**Meme Application**

**A PROJECT REPORT**

**Submitted By**

**Ashutosh Sharma**

**1900290149027**

**Submitted in partial fulfillment of the**

**Requirements for the degree of**

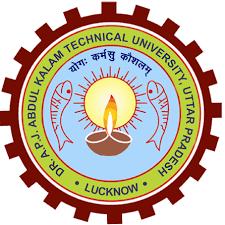
**MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**

**Ms. Neelam Rawat**

**ASSOCIATE PROFESSOR**

**KIET Group of Institutions, Ghaziabad**

****

**Submitted to**

**DEPARTMENT OF COMPUTER APPLICATION**

**Dr. A.P.J. Abdul Kalam Technical University**

**Uttar Pradesh-226031**

**(JULY 2021)**



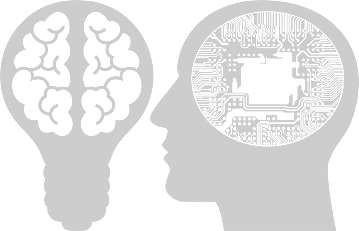
**Relieving Cum Experience Letter**

Date – May 27,2021

Name- Ashutosh Sharma

**To Whomsoever It May Concern**

This is to certify that **Ashutosh Sharma** has worked with us as an ‘’Android Developer Trainee’’ from 4/Jan/2021 to 07/May/2021. He has been relieved from his duties from 07/May/2021.



During his t

raining

with us, we found him hardworking, diligent and honest in performing his

duties.

The manag

ement would like to thank you for your services with Zimozi solutions and we

wish you all the best in your future endeavors.

Best Regards,

**Priyanka Bijolia**

**HR HEAD**



**Zimozi Solutions Pvt. Ltd.**

**DECLARATION**

I hereby declare that the work presented in this report entitled “Meme Application", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Ashutosh Sharma

1900290149027

Branch: Master of Computer Application

**(Candidate Signature)**

**CERTIFICATE**

Certified that **Ashutosh Sharma (1900290149027)** have carried out the project work having “**Meme Application**” for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Ms.Neelam Rawat External Examiner**

Associate Professor

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

**Dr. Ajay kumar Shrivastava**

Professor & Head,

Department of Computer Applications KIET Group of Institutions, Ghaziabad

**ABSTRACT**

**“Meme Application” is based on android application** more commonly known simply as a Meme is an idea, behaviour or style(meme) that is spread via the Internet, often through Social Media platform and especially for humorous purposes. What is considered a meme may vary across different communities on the Internet and is subject to change over time. Traditionally, they were a concept or catchphrase, but the concept has since became broader and more multi-faceted, evolving to include more elaborate structures such as Challenges GIF, videos, and Viral Internet memes are considered a part of Internet Culture they can spread from person to person via social networks, blogs, direct email, or new sources. Instant communication on the Internet facilitates word of mouth, transmission, resulting in fads and sensation that tend to grow rapidly. An example of such a fad is that of planking (lying down in public places);posting a photo of someone planking online brings attention to the fad and allows it to reach many people in little time Memes and social media are perfect for each other expressing a culturally- relevant idea.

A meme is an image or video that represents the thoughts and feelings of a specific audience.

**ACKNOWLEDGEMENTS**

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Ms. Neelam Rawat** for her guidance, help and encouragement throughout my research work.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Ashutosh Sharma**

**TABLE OF CONTENT**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Declaration | iii | |
|  | Certificate | iv | |
|  | Abstract | v | |
|  | Acknowledgements | vi | |
|  | Table of contents | viii | |
|  |  |  | |
| **CHAPTER 1** | **Introduction ix-x** |  | |
|  | 1.1 Project details |  | |
|  | 1.2 Project Purpose |  | |
|  | 1.3 effective |  | |
|  | 1.4 Android |  | |
| **CHAPTER 2** | Android Studio 11-14 |  | |
|  | 2.1 About | |
|  | 2.2 Feature |  | |
|  | 2,3 Android |  | |
| **CHAPTER 3** | **System Design 15-20** |  | |
|  |  |  | |
|  | 3.1 Java |  | |
|  | 3.2 Java Versions |  | |
| **CHAPTER 4** | **Architecture Diagram of Meme 21-23**  4.1 Meme Generator Diagram  4.2 Performance  4.3 Portability Requirements  4.4 Delivery Requirements |  | |
| **CHAPTER 5** | **Attribute of the Software 24**  5.1 Maintainability  5.2 Adaptability  5.3 Availability  5.4 Flexibility |  | |
| **CHAPTER 6** | **Implementation Results 25-26** |  | |
|  | 6.1 Implementation Results  6.2 Meme Output from emulator |  | |
|  |  |  | |
| **CHAPTER 7** | **Software components and technology 27** |  | |
|  | 7.1 Technology Enablers |  | |
|  | 7.2 Operating System  7.3 Hardware Components |  | |

|  |  |
| --- | --- |
| **CHAPTER 8** | **Testing 28 -29**  8.1 Steeple Analysis  8.2 Economic  8.3 Political  8.4 Technological  8.5 Legal  8.6 Ethical  **Source Code 29-96**  **Conclusion 97**    **References 98-99**  **Bibliography 100-101** |
|  |  |

**CHAPTER 1**

**INTRODUCTION**

* 1. **Project Details**

**“Meme Application” is based on android application** by using Java Language and platform Android Studio it is more commonly known simply as a **meme** is an idea, behaviour, or style (meme) that is spread via the Internet, often through Social Media platform and especially for humorous purposes. What is considered a meme may vary across different communities on the Internet and is subject to change over time. Traditionally they were a concept or catchphrase, but the concept has since become broader and more multi-faceted, evolving to include more elaborate structures such as Challenges, GIF, videos, and Viral.

* 1. **Purpose**

Internet memes are considered a part of Internet Culture They can spread from person to person via social networks, blogs, direct email, or news sources. Instant communication on the Internet facilitates word of mouth, transmission, resulting in fads and sensations that tend to grow rapidly. An example of such a fad is that of planking (lying down in public places); posting a photo of someone planking online brings attention to the fad and allows it to ‘reach many people in little time. The Internet also facilitates the rapid evolution of memes.

**1.3 Effective**

An Internet meme may stay the same or may evolve over time, by chance or through commentary, imitations, parody, or by incorporating news accounts about itself. Internet memes spread online through influences such as popular culture.

**1.4 Android**

Android is a mobile operating system. based on a modified version of the Linux Kernel and other opensource software, designed primarily for mobile devices such as Smartphones, Tablets etc.

About 70 percent of Android smartphones run Google's ecosystem; some with vendor-customized user interface and software suite, such as *TouchWiz* and later *One UI* by Samsung, *and*  Competing Android ecosystems include Fire OS (developed by Amazon) or However the "Android" name and logo are trademarks of Google which impose standards to restrict "uncertified" devices outside their ecosystem to use Android brand.

**1.5 Scope:**

Memes and social media are perfect for each other. Both concepts are about connecting with people online. While social media is a platform for communication, memes are a way of expressing a culturally-relevant idea. A meme is an image or video that represents the thoughts and feelings of a specific audience. Most memes are captioned photos intended to elicit humour. However, there are many viral video memes too.

Memes are a worldwide social phenomenon, and an increasingly important aspect of viral marketing and social engagement. Memes often relate to existing cultures or subcultures. Often, memes spread rapidly through social media, email, and forum board Though memes are popular in the consumer world, they’re also powerful for companies too. Companies use memes to:

* Share information in a fun and memorable format.
* Making Innovative memes
* Humanize your brand through a demonstration of personality.
* Funny and humours
* Showcase complicated concepts like company culture uniquely.
* Using stickers
* Obtain higher engagement with their audience.

**CHAPTER 2**

**Android Studio**

**2.1 About:**

**Android Studio** is the official Integrated Development Environment (IDE) for Google's Android Operating System, built on JetBrains software and designed specifically for Android Development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.

**2.2 Feature:**

A specific feature of the Android Studio is an absence of the possibility to switch autosave feature off.

The following features are provided in the current stable version:

* Gradle-based build support
* Android-specific and quick fixes
* Lint tools to catch performance, usability, version compatibility and other problems
* Pro Guard integration and app-signing capabilities
* Template-based wizards to create common Android designs and components
* A rich layout editor that allows users to drag-and-drop UI components, option to preview  on multiple screen configurations
* Support for building Android Wear apps
* Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine.

Android Studio supports all the same programming languages of [IntelliJ](https://en.wikipedia.org/wiki/IntelliJ) e.g. java ,C++ and more with extensions, such and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android. Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy.

The Android Emulator has additional requirements beyond the basic system requirements for Android Studio, which are described below:

* SDK Tools 26.1.1 or higher;
* 64-bit processor;
* Windows: CPU with UG (unrestricted guest) support;
* Intel Hardware Accelerated Execution Manager (**HAXM**) 6.2.1 or later (HAXM 7.2.0 or later recommended).

The use of hardware acceleration has additional requirements on Windows and Linux:

* Intel processor on Windows or Linux: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality.
* AMD processor on Linux: AMD processor with support for AMD Virtualization (AMD-V) and [Supplemental Streaming SIMD Extensions 3 (SSSE3)](https://en.wikipedia.org/wiki/SSSE3);
* AMD processor on Windows: Android Studio 3.2 or higher and Windows 10 April 2018 release or higher for [Windows Hypervisor Platform (WHPX)](https://docs.microsoft.com/en-us/virtualization/api/) functionality.

To work with Android 8.1 (API level 27) and higher system images, an attached

webcam must have the capability to capture 720p frames.

**ANDROID**

**Android** is a mobile operating system based on a modified version of the Linux Kernel and other opensource software, designed primarily for touch screen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google. It was unveiled in November 2007, with the first commercial Android device, the HTC Dream. being launched in September 2008.It is free and opensource Software; its source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache license. However most Android devices ship with additional pre-installed, most notably Google Mobile Service (GMS)[[13]](https://en.wikipedia.org/wiki/Android_(operating_system)#cite_note-13) which includes core apps such as Google Chrome, the digital distribution platform Google Play and associated Google Play Services development platform About 70 percent of Android smartphones run Google's ecosystem; some with vendor-customized user interface and software suite, such as *TouchWiz* and later *One UI* by Samsung, and *HTC Sense*. Competing Android ecosystems and include Fire-OS (developed by Amazon) or . However the "Android" name and logo are trademarks of Google which impose standards to restrict "uncertified" devices outside their ecosystem to use Android branding. The source code has been used to develop variants of Android on a range of other electronics, such as game consoles, digital cameras, portable media players, PCs and others, each with a specialized user interface. Some well known derivatives include Android Tv for televisions and Wear OS for wearables, both developed by Google. Software packages on Android, which use the APK format, are generally distributed through proprietary Application stores like Google play Stores, Samsung galaxy stores App Gallery,, and Get Jar, or open source platforms like or F-droid Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2021, it has over three billion monthly active services, the largest of any operating system, and as of January 2021, the Google Play Store features over 3 million apps. The current stable version is Android 11 released on September 8, 2020.

**CHAPTER 3**

**JAVA**

**3.1 Introduction**

**Java** is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general -purpose programming language intended to let application developers  *write once, run anywhere* (WORA),[[16]](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_note-16) meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of the underlying computer Architecture. The syntax of java is similar to C and C++, but has fewer Low Level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages. As of 2019, Java was one of the most popular programming language In use according to Git Hub, particularly for client-server Web Applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (Which has been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java Platform. The original and reference Implementations Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary. As of May 2007, in compliance with the specifications of the Java Community process, Sun had relicensed most of its Java technologies under the license. Oracle offers its own Hot Spot Java Virtual Machine, however the official reference implementation is the OpenJDK JVM which is free open source software and used by most developers and is the default JVM for almost all Linux distributions. As of March 2021, the latest version is Java 16, with Java 11, a currently supported Long term Service (LTS) version, released on September 25, 2018. Oracle released the last zero-cost public update for the legacy version Java 8 LTS in January 2019 for commercial use, although it will otherwise still support Java 8 with public updates for personal use indefinitely. Other vendors have begun to offer Zero cost builds of OpenJDK 8 and 11 that are still receiving security and other upgrades Oracle (and others) highly recommend uninstalling outdated versions of Java because of serious risks due to unresolved security issues.[[21]](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_note-21) Since Java 9, 10, 12, 13, 14, and 15 are no longer supported, Oracle advises its users to immediately transition to the latest version (currently Java 16) or an LTS release.

**3.2 Java Versions**

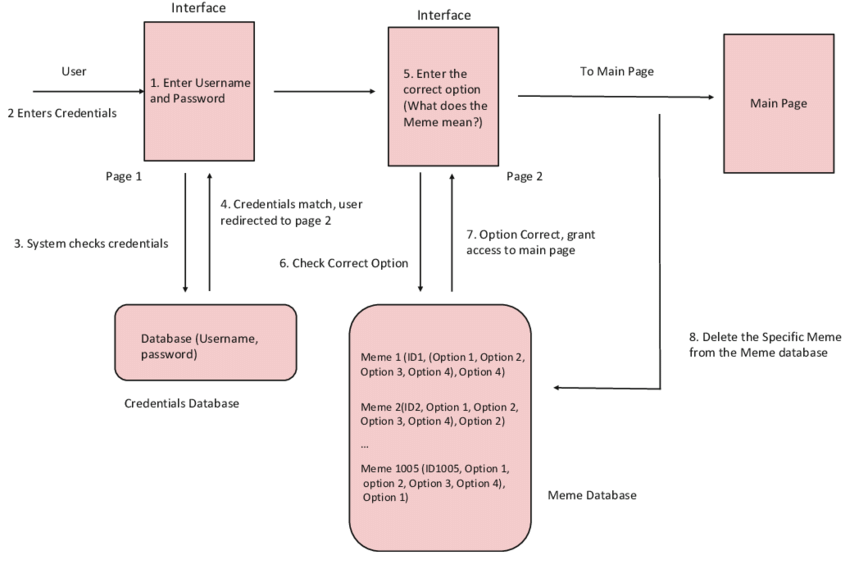
The Java Language has undergone several changes since JDK 1.0 as well as numerous additions of classes and packages to the standard library. Since J2SE 1.4, the evolution of the Java language has been governed by the Java Community Process (JCP), which uses *Java Specification Requests* (JSRs) to propose and specify additions and changes to the Java Platform. The language is specified by the *Java Language Specification* (JLS); changes to the JLS are managed under JSR 901.In addition to the language changes, other changes have been made to the Java Class Library over the years, which has grown from a few hundred classes in JDK 1.0 to over three thousand in J2SE 5. Entire new APIs, such as Swing and java2D, have been introduced, and many of the original JDK 1.0 classes and methods have been deprecated. Some programs allow conversion of Java programs from one version of the java platform to an older one (for example Java 5.0 backported to 1.4) (see java backporting tool).Regarding Oracle Java SE Support Roadmap,[[1]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-auto9-1) version 11 is the currently supported Long Term Support (LTS) version, together with Java 8 LTS, where Oracle Customers will receive Oracle Premier Support. Java 8 LTS last free software public update for commercial use was released by Oracle in January 2019, while Oracle continues to release no-cost public Java 8 updates for development[[1]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-auto9-1) and personal use indefinitely.[[2]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-Java8-2) Java 10 a previously supported rapid release version, had its support ended in September 2018 the same date support for Java 11 began. Java 7 is no longer publicly supported. For Java 11, long-term support will not be provided by Oracle for the public; instead, the broader OpenJDK community, as AdoptOpenJdk or others, is expected to perform the work.[[3]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-OracleJava11LTS-3)Java 16 General Availability occurred on March 16, 2021, with Java 17 .

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Release date** | **End of Free Public Updates**[[1]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-auto9-1)[[6]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-6)[[7]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-7)[[8]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-8) | **Extended Support Until** |
| JDK Beta | 1995 | ? | ? |
| JDK 1.0 | January 1996 | ? | ? |
| JDK 1.1 | February 1997 | ? | ? |
| J2SE 1.2 | December 1998 | ? | ? |
| J2SE 1.3 | May 2000 | ? | ? |
| J2SE 1.4 | February 2002 | October 2008 | February 2013 |
| J2SE 5.0 | September 2004 | November 2009 | April 2015 |
| Java SE 6 | December 2006 | April 2013 | December 2018 December 2026, paid support for Azul Platform Core[[9]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-Azul-9) |
| Java SE 7 | July 2011 | April 2015 | July 2022 |
| Java SE 8 (LTS) | March 2014 | **January 2019 for Oracle (commercial)** December 2030 for Oracle (non-commercial) December 2030 for Azul At least May 2026 for AdoptOpenJDK At least May 2026 for Amazon Corretto | December 2030 |
| Java SE 9 | September 2017 | March 2018 for OpenJDK | N/A |
| Java SE 10 | March 2018 | September 2018 for OpenJDK | N/A |
| Java SE 11 (LTS) | September 2018 | September 2026 for Azul At least October 2024 for AdoptOpenJDK At least September 2027 for Amazon Corretto | September 2026, or September 2028 for Azul[[9]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-Azul-9) |
| Java SE 12 | March 2019 | September 2019 for OpenJDK | N/A |
| Java SE 13 | September 2019 | March 2020 for OpenJDK | N/A |
| Java SE 14 | March 2020 | September 2020 for OpenJDK | N/A |
| Java SE 15 | September 2020 | March 2021 for OpenJDK, March 2023 for Azul[[9]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-Azul-9) | N/A |
| **Java SE 16** | March 2021 | September 2021 for OpenJDK | N/A |
| Java SE 17 (LTS) | September 2021 | September 2030 for Zulu | TBA |
| Java SE 18 | March 2022 | September 2022 for OpenJDK | N/A |

|  |  |
| --- | --- |
| **Version** | **Date** |
| JDK Beta | 1995 |
| JDK1.0 | January 23, 1996[[39]](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_note-39) |
| JDK 1.1 | February 19, 1997 |
| J2SE 1.2 | December 8, 1998 |
| J2SE 1.3 | May 8, 2000 |
| J2SE 1.4 | February 6, 2002 |
| J2SE 5.0 | September 30, 2004 |
| Java SE 6 | December 11, 2006 |
| Java SE 7 | July 28, 2011 |
| Java SE 8 | March 18, 2014 |
| Java SE 9 | September 21, 2017 |
| Java SE 10 | March 20, 2018 |
| Java SE 11 | September 25, 2018[[40]](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_note-40) |
| Java SE 12 | March 19, 2019 |
| Java SE 13 | September 17, 2019 |
| Java SE 14 | March 17, 2020 |
| Java SE 15 | September 15, 2020[[41]](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_note-41) |
| Java SE 16 | March 16, 2021 |

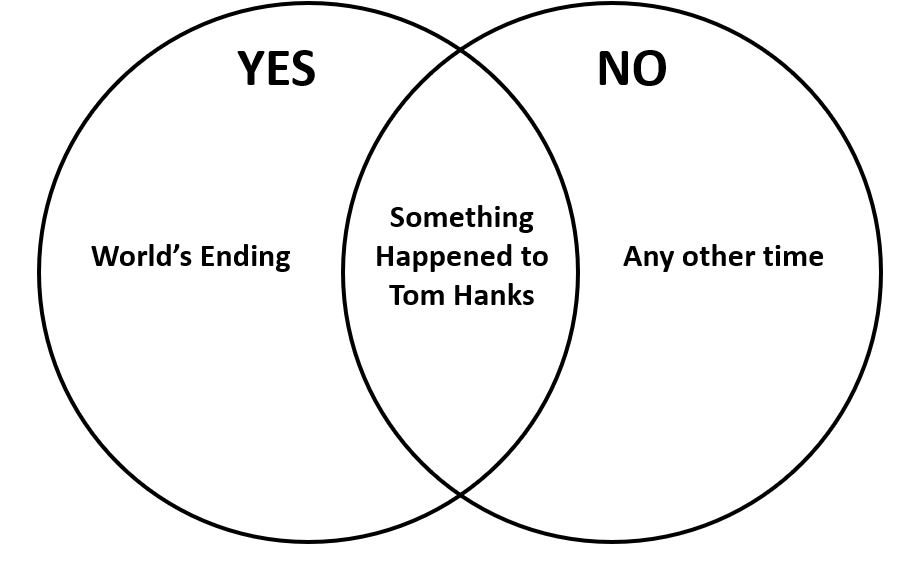
**Chapter 4**

**Architecture Diagram Of Meme**



**Figure 1ER Diagram**

**4.1 Meme Generator Diagram**



**Figure 2 Meme Generator Diagram**

**4.2 Performance**

The application response time shall be adequate and sufficient enough, that's why the time required for this application to response to its user's actions has to been managed and controlled. But in order to maintain the performance of the application, the user has to follow the required steps to get the desired result.

**4.3. Portability Requirements:**

The application should be compatible with different version of Android, so if the version of Android is upgraded, the application should be upgraded as well.

**4.3. Delivery Requirements:**

I agreed with my client, to deliver the mobile based application by the beginning of December, and exactly on December 4th, 2015.

**4.4. Implementation Requirements:**

I used Java as a programming language for the implementation of the project

**4.5. Standards Requirements:**

The application shall follow the AUI standard form.

**4.6. Ethical Requirements:**

This application should protect the confidentiality of the user's personal information and any personal data stored on his\her mobile phone.

**4.7. Security**:

The security signature and certificate of the application is required as in any mobile application.

**4.8. Privacy:**

The application shall protect the user's data and make sure to keep it confidential. The device can be protected by a pin code or finger prints in order to ensure the privacy.

**CHAPTER 5**

**Attributes of the Software:**

**5.1. Maintainability:**

The application shall respond to any change on the requirements.

**5.2. Adaptability:**

The application shall be compatible to any Android OS version.

**5.3. Availability:**

The application shall be available on the store whenever users want to

download it.

**5.4. Flexibility:**

The architecture shall be flexible to any change of the requirements.

**CHAPTER 6**

**6.1 Implementation Results**

The implementation results shows the end result of the project. It is an Android based device application that fulfills the requirements set by the client. These are some snapshots of the application with their description.



Figure 3 Implementaion Result

Figure 4 Implementation Result

**6.2 Meme Output From Emulator**

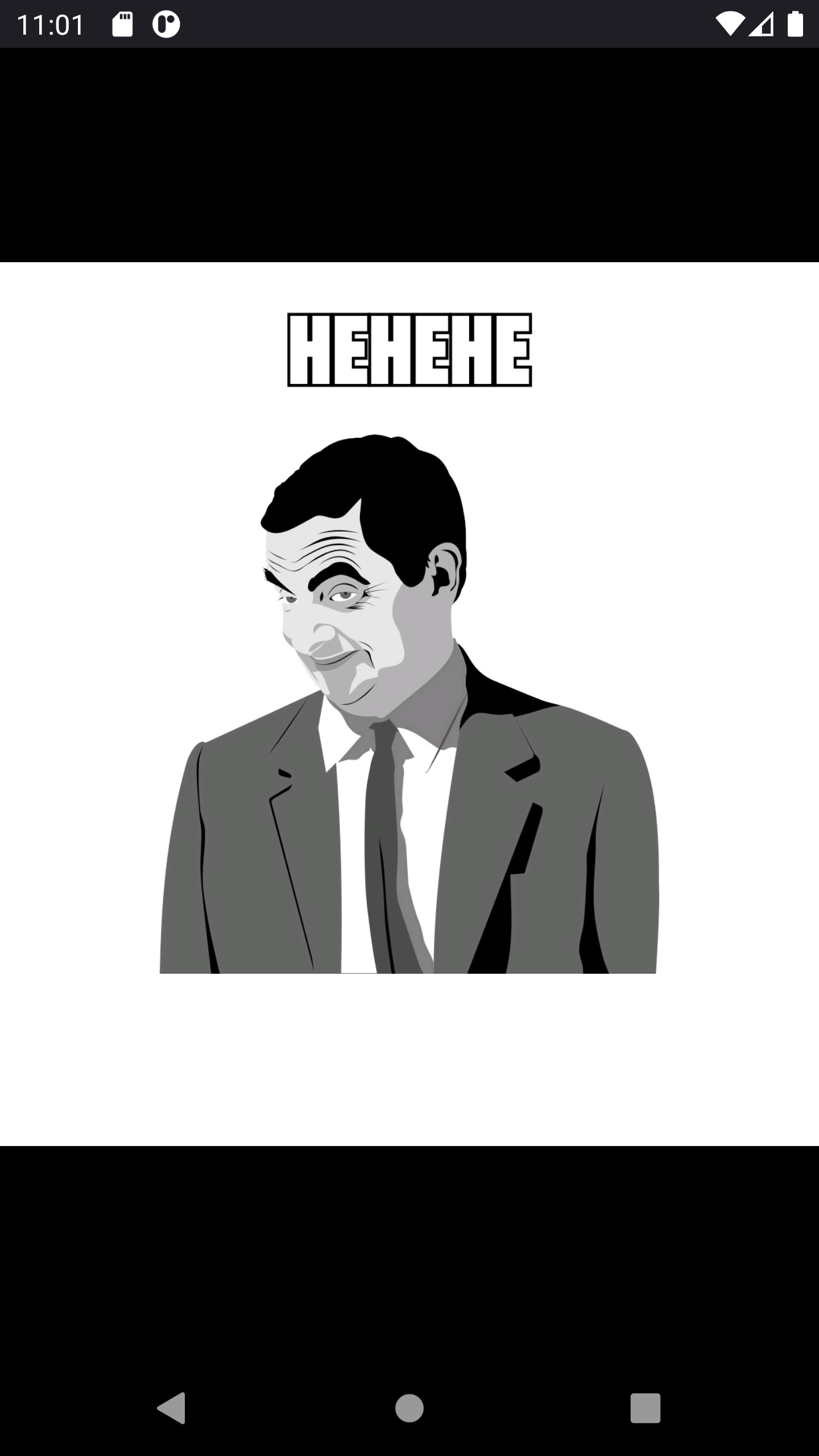


Figure 5 Meme Output

**CHAPTER 7**

**Software Components and Technology used:**

The software components and technology used in this project are:

**7.1. Technology enablers:**

* Java as an object oriented programming language.
* Android SDK( Software development Kit)
* Android Studio.

**7.2. Operating Systems:**

* Windows
* Android OS

**7.3. Hardware Components:**

* Personal Computers
* Android based mobile phone with high resolution.

**CHAPTER 8**

**Testing:**

Testing is an important step that helps to detect errors. Testing is a process of finding faults that might occur during the implementation phase. It is also a way to test if the product fulfills the requirements and to check the components functionalities. There exists many ways of testing where each one of them has a distinct requirement, but the only testing that we made is the acceptance testing. I have tested the application, Android based device, with the supervisor using acceptance testing strategy.

**8.1. STEEPLE Analysis:**

The STEEPLE Analysis includes seven external factors that can affect or be affected by the industry. Actually, before moving to the step of the implementation, we must know the risks and challenges that we might face later on. In other words, we must analyze the possible chances and threats to the industry as a whole.

**8.2.Economic**

Since people will be using a such technology, so instead of wastingtime typing on their keyboards or writing on papers, many actions might be performed in few minutes which will contribute on the growth of the economy.

**8.3. Political:** No political trend.

**8.4. Technological:** This application helps to Create Funny , Innovative memes.

**8.5. Legal:** No legal trend.

**8.6. Ethical:** This application should protect the confidentiality of the user'spersonal information and any personal data stored on the mobile phone.

**SOURCE CODE**

**package net.gsantner.memetastic.activity;  
  
import android.graphics.Bitmap;  
import android.graphics.Color;  
import android.os.Build;  
import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.support.v4.app.FragmentManager;  
import android.support.v4.app.FragmentStatePagerAdapter;  
import android.support.v4.view.ViewPager;  
import android.support.v7.app.ActionBar;  
import android.support.v7.app.AppCompatActivity;  
import android.support.v7.widget.Toolbar;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.Window;  
import android.view.WindowManager;  
  
import net.gsantner.memetastic.App;  
import net.gsantner.memetastic.data.MemeData;  
import net.gsantner.memetastic.service.AssetUpdater;  
import net.gsantner.memetastic.util.AppSettings;  
import net.gsantner.memetastic.util.PermissionChecker;  
  
import java.io.File;  
import java.util.List;  
  
import butterknife.BindView;  
import butterknife.ButterKnife;  
import io.github.gsantner.memetastic.R;  
  
public class ImageViewActivity extends AppCompatActivity {  
 *//########################  
 //## UI Binding  
 //########################* @BindView(R.id.*imageview\_activity\_\_view\_pager*)  
 ViewPager \_viewPager;  
  
 @BindView(R.id.*toolbar*)  
 Toolbar \_toolbar;  
  
 *//#####################  
 //## Members  
 //#####################* private File \_imageFile;  
 private Bitmap \_bitmap = null;  
 List<MemeData.Image> imageList = null;  
  
 *//#####################  
 //## Methods  
 //#####################* @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 if (PermissionChecker.*hasExtStoragePerm*(this)) {  
 File folder = AssetUpdater.*getMemesDir*(AppSettings.*get*());  
 folder.mkdirs();  
 imageList = MemeData.*getCreatedMemes*();** **}  
  
 if (AppSettings.*get*().isOverviewStatusBarHidden()) {  
 getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*, WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);  
 }  
 setContentView(R.layout.*imageview\_\_activity*);  
 ButterKnife.*bind*(this);  
 *//ContextUtils.get().enableImmersiveMode(getWindow().getDecorView());* setSupportActionBar(\_toolbar);  
 if (getSupportActionBar() != null) {  
 ActionBar ab = getSupportActionBar();  
 ab.setDisplayShowTitleEnabled(false);  
 ab.setDisplayHomeAsUpEnabled(true);  
 }  
  
 \_viewPager.setAdapter(new ImagePagerAdapter(getSupportFragmentManager()));  
 \_viewPager.setCurrentItem(getIntent().getIntExtra(MainActivity.*IMAGE\_POS*, 0));  
  
  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*LOLLIPOP*) {  
 Window window = getWindow();  
 window.addFlags(WindowManager.LayoutParams.*FLAG\_DRAWS\_SYSTEM\_BAR\_BACKGROUNDS*);  
 window.setStatusBarColor(Color.*BLACK*);  
 }  
  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*imageview\_\_menu*, menu);  
 *// menu.findItem(R.id.action\_delete).setVisible(\_imageFile != null);* return true;  
 }  
  
 @Override  
 public void onBackPressed() {  
 finish();  
 }  
  
 @Override  
 protected void onDestroy() {  
*// \_expandedImageView.setImageBitmap(null);  
// if (\_bitmap != null && !\_bitmap.isRecycled())  
// \_bitmap.recycle();* super.onDestroy();  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 ImageViewFragment page = null;  
  
 if (item.getItemId() == R.id.*action\_share* || item.getItemId() == R.id.*action\_delete*) {  
 page = ((ImageViewFragment) \_viewPager.getAdapter().instantiateItem(\_viewPager, \_viewPager.getCurrentItem()));  
 }  
 switch (item.getItemId()) {  
  
 case android.R.id.*home*: {  
 finish();  
 return true;  
 }  
 case R.id.*action\_share*: {  
 if (page != null) {  
 \_bitmap = page.\_bitmap;  
 ((App) getApplication()).shareBitmapToOtherApp(\_bitmap, this);  
 }  
 return true;  
 }  
  
 case R.id.*action\_delete*: {  
  
 \_imageFile = page.\_imageFile;  
  
 if (\_imageFile != null) {  
 deleteFile(\_imageFile);  
 deleteFile(new File(getCacheDir(), \_imageFile.getAbsolutePath().substring(1)));  
 MemeData.Image memeData = MemeData.*findImage*(\_imageFile);  
 if (memeData != null) {  
 MemeData.*getCreatedMemes*().remove(memeData);  
 }  
 }  
 \_viewPager.getAdapter().notifyDataSetChanged();  
 finish();  
 return true;  
 }  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 private boolean deleteFile(File file) {  
 return file.exists() && file.delete();  
 }  
  
  
 */\*\*  
 \* The conf was clicked  
 \*/* class ImagePagerAdapter extends FragmentStatePagerAdapter {  
  
 public ImagePagerAdapter(FragmentManager fm) {  
 super(fm);  
  
 }  
  
 @Override  
 public Fragment getItem(int i) {  
 return ImageViewFragment.*newInstance*(i, imageList.get(i).fullPath.getAbsolutePath());  
 }  
  
 @Override  
 public int getCount() {  
 return imageList.size();  
 }  
 }  
}**

**ImageViewFragment**

**package net.gsantner.memetastic.activity;  
  
  
import android.graphics.Bitmap;  
import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.support.v4.content.ContextCompat;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
  
import net.gsantner.memetastic.util.ContextUtils;  
import net.gsantner.memetastic.util.PermissionChecker;  
  
import java.io.File;  
  
import butterknife.BindView;  
import butterknife.ButterKnife;  
import butterknife.OnClick;  
import io.github.gsantner.memetastic.R;  
  
  
public class ImageViewFragment extends Fragment {  
  
  
 @BindView(R.id.*imageview\_fragment\_\_expanded\_image*)  
 ImageView \_expandedImageView;  
  
  
 private static final String *ARG\_PARAM\_\_POS* = "pos";  
 private static final String *ARG\_PARAM\_\_IMAGE\_PATH* = "param2";  
  
  
 private int \_position;  
 public String \_imagePath;  
 public File \_imageFile;  
  
 public Bitmap \_bitmap;  
  
  
 public ImageViewFragment() {  
 *// Required empty public constructor* }  
  
  
 public static ImageViewFragment newInstance(int position, String param2) {  
 ImageViewFragment fragment = new ImageViewFragment();  
 Bundle args = new Bundle();  
 args.putInt(*ARG\_PARAM\_\_POS*, position);  
 args.putString(*ARG\_PARAM\_\_IMAGE\_PATH*, param2);  
 fragment.setArguments(args);  
 return fragment;  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 if (getArguments() != null) {  
 \_position = getArguments().getInt(*ARG\_PARAM\_\_POS*);  
 \_imagePath = getArguments().getString(*ARG\_PARAM\_\_IMAGE\_PATH*);  
 }  
  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 *// Inflate the layout for this fragment* View view = inflater.inflate(R.layout.*fragment\_image\_view*, container, false);  
  
 ButterKnife.*bind*(this, view);  
  
  
 \_imageFile = new File(\_imagePath);  
 if (PermissionChecker.*hasExtStoragePerm*(getActivity()) && \_imageFile.exists()) {  
 \_bitmap = ContextUtils.*get*().loadImageFromFilesystem(\_imageFile);  
 }  
 if (\_bitmap == null) {  
 \_imageFile = null;  
 \_bitmap = ContextUtils.*get*().drawableToBitmap(  
 ContextCompat.*getDrawable*(getActivity(), R.drawable.*ic\_mood\_bad\_black\_256dp*));  
 }  
 \_expandedImageView.setImageBitmap(\_bitmap);  
  
 return view;  
 }  
  
 @Override  
 public void onDestroy() {  
 \_expandedImageView.setImageBitmap(null);  
 if (\_bitmap != null && !\_bitmap.isRecycled())  
 \_bitmap.recycle();  
 super.onDestroy();  
 }  
  
 @OnClick(R.id.*imageview\_fragment\_\_expanded\_image*)  
 public void onImageClicked() {  
 getActivity().finish();  
 }  
  
}**

**MAINACTIVITY**

**package net.gsantner.memetastic.activity;  
  
import android.annotation.SuppressLint;  
import android.app.SearchManager;  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.database.Cursor;  
import android.graphics.PorterDuff;  
import android.net.Uri;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Environment;  
import android.os.Handler;  
import android.os.ParcelFileDescriptor;  
import android.provider.MediaStore;  
import android.support.annotation.DrawableRes;  
import android.support.annotation.NonNull;  
import android.support.annotation.StringRes;  
import android.support.design.widget.BottomNavigationView;  
import android.support.design.widget.NavigationView;  
import android.support.design.widget.TabLayout;  
import android.support.v4.content.ContextCompat;  
import android.support.v4.content.FileProvider;  
import android.support.v4.content.LocalBroadcastManager;  
import android.support.v4.view.ViewPager;  
import android.support.v7.app.AlertDialog;  
import android.support.v7.app.AppCompatActivity;  
import android.support.v7.widget.GridLayoutManager;  
import android.support.v7.widget.LinearLayoutManager;  
import android.support.v7.widget.RecyclerView;  
import android.support.v7.widget.SearchView;  
import android.support.v7.widget.Toolbar;  
import android.text.TextUtils;  
import android.util.Base64;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.WindowManager;  
import android.widget.FrameLayout;  
import android.widget.ImageView;  
import android.widget.LinearLayout;  
import android.widget.ProgressBar;  
import android.widget.TextView;  
  
import net.gsantner.memetastic.App;  
import net.gsantner.memetastic.data.MemeData;  
import net.gsantner.memetastic.service.AssetUpdater;  
import net.gsantner.memetastic.ui.GridDecoration;  
import net.gsantner.memetastic.ui.MemeItemAdapter;  
import net.gsantner.memetastic.util.ActivityUtils;  
import net.gsantner.memetastic.util.AppCast;  
import net.gsantner.memetastic.util.AppSettings;  
import net.gsantner.memetastic.util.ContextUtils;  
import net.gsantner.memetastic.util.PermissionChecker;  
import net.gsantner.opoc.format.markdown.SimpleMarkdownParser;  
import net.gsantner.opoc.ui.LinearSplitLayout;  
import net.gsantner.opoc.util.AndroidSupportMeWrapper;  
import net.gsantner.opoc.util.FileUtils;  
  
import java.io.File;  
import java.io.FileDescriptor;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.lang.reflect.Method;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
import butterknife.BindView;  
import butterknife.ButterKnife;  
import io.github.gsantner.memetastic.R;  
  
public class MainActivity extends AppCompatActivity  
 implements NavigationView.OnNavigationItemSelectedListener, ViewPager.OnPageChangeListener, BottomNavigationView.OnNavigationItemSelectedListener {  
 public static final int *REQUEST\_LOAD\_GALLERY\_IMAGE* = 50;  
 public static final int *REQUEST\_TAKE\_CAMERA\_PICTURE* = 51;  
 public static final int *REQUEST\_SHOW\_IMAGE* = 52;  
 public static final String *IMAGE\_PATH* = "imagePath";  
 public static final String *IMAGE\_POS* = "image\_pos";  
 public static final boolean *LOCAL\_ONLY\_MODE* = true;  
 public static final boolean *DISABLE\_ONLINE\_ASSETS* = true;  
  
 private static boolean *\_isShowingFullscreenImage* = false;  
  
 @BindView(R.id.*toolbar*)  
 Toolbar \_toolbar;  
  
 @BindView(R.id.*bottom\_navigation\_bar*)  
 BottomNavigationView \_bottomNav;  
 private MenuItem \_lastBottomMenuItem;  
  
 @BindView(R.id.*main\_\_tabs*)  
 TabLayout \_tabLayout;  
  
 @BindView(R.id.*main\_\_more\_info\_fragment\_container*)  
 LinearLayout \_moreInfoContainer;  
  
 @BindView(R.id.*main\_activity\_\_placeholder*)  
 FrameLayout \_placeholder;  
  
 @BindView(R.id.*main\_activity\_\_view\_pager*)  
 ViewPager \_viewPager;  
  
 @BindView(R.id.*main\_\_activity\_\_recycler\_view*)  
 RecyclerView \_recyclerMemeList;  
  
 @BindView(R.id.*main\_\_activity\_\_list\_empty\_\_layout*)  
 LinearSplitLayout \_emptylistLayout;  
  
 @BindView(R.id.*main\_\_activity\_\_list\_empty\_\_text*)  
 TextView \_emptylistText;  
  
 @BindView(R.id.*main\_\_activity\_\_infobar*)  
 LinearLayout \_infoBar;  
  
 @BindView(R.id.*main\_\_activity\_\_infobar\_\_progress*)  
 ProgressBar \_infoBarProgressBar;  
  
 @BindView(R.id.*main\_\_activity\_\_infobar\_\_image*)  
 ImageView \_infoBarImage;  
  
 @BindView(R.id.*main\_\_activity\_\_infobar\_\_text*)  
 TextView \_infoBarText;  
  
 App app;  
 private AppSettings \_appSettings;  
 private ActivityUtils \_activityUtils;  
 private String cameraPictureFilepath = "";  
 String[] \_tagKeys, \_tagValues;  
 private int \_currentMainMode = 0;  
 private long \_lastInfoBarTextShownAt = 0;  
 private SearchView \_searchView;  
 private MenuItem \_searchItem;  
 private String \_currentSearch = "";  
  
 private static final String *BOTTOM\_NAV\_POSITION* = "bottom\_nav\_position";  
  
 @SuppressLint("ClickableViewAccessibility")  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 if (savedInstanceState != null) {  
 \_currentMainMode = savedInstanceState.getInt(*BOTTOM\_NAV\_POSITION*);  
 }  
 \_appSettings = new AppSettings(this);  
 \_activityUtils = new ActivityUtils(this);  
 \_activityUtils.setAppLanguage(\_appSettings.getLanguage());  
 if (\_appSettings.isOverviewStatusBarHidden()) {  
 getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*, WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);  
 }  
 setContentView(R.layout.*main\_\_activity*);  
  
 *// Bind UI* app = (App) getApplication();  
 ButterKnife.*bind*(this);  
  
 *// Setup \_toolbar* setSupportActionBar(\_toolbar);  
  
 \_tagKeys = getResources().getStringArray(R.array.*meme\_tags\_\_keys*);  
 \_tagValues = getResources().getStringArray(R.array.*meme\_tags\_\_titles*);  
  
 if (MainActivity.*LOCAL\_ONLY\_MODE*) {  
 for (int i = 0; i < \_tagKeys.length; i++) {  
 \_tagKeys[i] = "other";  
 }  
 \_tagKeys = new String[]{\_tagKeys[0]};  
 \_tagValues = new String[]{\_tagValues[0]};  
 }  
  
  
 \_recyclerMemeList.setHasFixedSize(true);  
 \_recyclerMemeList.setItemViewCacheSize(\_appSettings.getGridColumnCountPortrait() \* \_appSettings.getGridColumnCountLandscape() \* 2);  
 \_recyclerMemeList.setDrawingCacheEnabled(true);  
 \_recyclerMemeList.setDrawingCacheQuality(View.*DRAWING\_CACHE\_QUALITY\_LOW*);  
 \_recyclerMemeList.addItemDecoration(new GridDecoration(1.7f));  
  
 if (\_appSettings.getMemeListViewType() == MemeItemAdapter.*VIEW\_TYPE\_\_ROWS\_WITH\_TITLE*) {  
 RecyclerView.LayoutManager recyclerLinearLayout = new LinearLayoutManager(this, LinearLayoutManager.*VERTICAL*, false);  
 \_recyclerMemeList.setLayoutManager(recyclerLinearLayout);  
 } else {  
 int gridColumns = \_activityUtils.isInPortraitMode()  
 ? \_appSettings.getGridColumnCountPortrait()  
 : \_appSettings.getGridColumnCountLandscape();  
 RecyclerView.LayoutManager recyclerGridLayout = new GridLayoutManager(this, gridColumns);  
  
 \_recyclerMemeList.setLayoutManager(recyclerGridLayout);  
 }  
  
 for (String cat : \_tagValues) {  
 TabLayout.Tab tab = \_tabLayout.newTab();  
 tab.setText(cat);  
 \_tabLayout.addTab(tab);  
 }  
  
 *// Basically enable "other" only mode* if (MainActivity.*LOCAL\_ONLY\_MODE*) {  
 \_tabLayout.setVisibility(View.*GONE*);  
 }  
 *// END* \_viewPager.setOffscreenPageLimit(5);  
 \_viewPager.setAdapter(new MemePagerAdapter(getSupportFragmentManager(), \_tagKeys.length, \_tagValues));  
 \_tabLayout.setupWithViewPager(\_viewPager);  
 selectTab(app.settings.getLastSelectedTab(), app.settings.getDefaultMainMode());  
 \_infoBarProgressBar.getProgressDrawable().setColorFilter(ContextCompat.*getColor*(this, R.color.*accent*), PorterDuff.Mode.*SRC\_IN*);  
  
  
 *// Show first start dialog / changelog* try {  
 if (\_appSettings.isAppCurrentVersionFirstStart(true)) {  
 SimpleMarkdownParser smp = SimpleMarkdownParser.*get*().setDefaultSmpFilter(SimpleMarkdownParser.*FILTER\_ANDROID\_TEXTVIEW*);  
 String html = "";  
 html += smp.parse(getString(R.string.*copyright\_license\_text\_official*).replace("\n", " \n"), "").getHtml();  
 html += "<br/><br/><br/><big><big>" + getString(R.string.*changelog*) + "</big></big><br/>" + smp.parse(getResources().openRawResource(R.raw.*changelog*), "", SimpleMarkdownParser.*FILTER\_ANDROID\_TEXTVIEW*, SimpleMarkdownParser.*FILTER\_CHANGELOG*);  
 html += "<br/><br/><br/><big><big>" + getString(R.string.*licenses*) + "</big></big><br/>" + smp.parse(getResources().openRawResource(R.raw.*licenses\_3rd\_party*), "").getHtml();  
  
 \_activityUtils.showDialogWithHtmlTextView(R.string.*licenses*, html);  
 }  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 new AssetUpdater.LoadAssetsThread(this).start();  
  
 if (PermissionChecker.*doIfPermissionGranted*(this)) {  
 ContextUtils.*checkForAssetUpdates*(this);  
 }  
  
 \_bottomNav.setOnNavigationItemSelectedListener(this);  
 }  
  
 public void updateHiddenNavOption() {  
 MenuItem hiddenItem = \_bottomNav.getMenu().findItem(R.id.*nav\_mode\_hidden*);  
 for (String hidden : app.settings.getHiddenMemesTemplate()) {  
 MemeData.Image image = MemeData.*findImage*(new File(hidden));  
 if (image != null) {  
 hiddenItem.setVisible(true);  
 return;  
 }  
 }  
 hiddenItem.setVisible(false);  
 }  
  
 @SuppressWarnings("ConstantConditions")  
 private void selectTab(int pos, int mainMode) {  
 MenuItem navItem = null;  
 switch (mainMode) {  
 case 0:  
 pos = pos >= 0 ? pos : \_tabLayout.getTabCount() - 1;  
 pos = pos < \_tabLayout.getTabCount() ? pos : 0;  
 \_tabLayout.getTabAt(pos).select();  
 break;  
 case 1:  
 navItem = \_bottomNav.getMenu().findItem(R.id.*nav\_mode\_favs*);  
 break;  
 case 2:  
 navItem = \_bottomNav.getMenu().findItem(R.id.*nav\_mode\_saved*);  
 break;  
 case 3:  
 navItem = \_bottomNav.getMenu().findItem(R.id.*nav\_mode\_hidden*);  
 break;  
 case 4:  
 navItem = \_bottomNav.getMenu().findItem(R.id.*nav\_more*);  
 break;  
 }  
  
 if (navItem != null) {  
 navItem.setChecked(true);  
 onNavigationItemSelected(navItem);  
 }  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 *//new AndroidSupportMeWrapper(this).mainOnResume();* if (MainActivity.*LOCAL\_ONLY\_MODE*) {  
 \_tabLayout.setVisibility(View.*GONE*);  
 }  
 if (*\_isShowingFullscreenImage*) {  
 *\_isShowingFullscreenImage* = false;  
 overridePendingTransition(R.anim.*fadein*, R.anim.*fadeout*);  
 }  
 LocalBroadcastManager.*getInstance*(this).registerReceiver(\_localBroadcastReceiver, AppCast.*getLocalBroadcastFilter*());  
  
 if (SettingsActivity.*activityRetVal* == SettingsActivity.RESULT.*CHANGE\_RESTART*) {  
 SettingsActivity.*activityRetVal* = SettingsActivity.RESULT.*NOCHANGE*;  
 recreate();  
 }  
  
 try {  
 if (new Random().nextInt(10) > 2) {  
 Method m = getClass().getMethod(new String(Base64.*decode*("Z2V0UGFja2FnZU5hbWU=", Base64.*DEFAULT*)));  
 String ret = (String) m.invoke(this);  
 if (!ret.startsWith(new String(Base64.*decode*("bmV0LmdzYW50bmVyLg==", Base64.*DEFAULT*))) && !ret.startsWith(new String(Base64.*decode*("aW8uZ2l0aHViLmdzYW50bmVyLg==", Base64.*DEFAULT*)))) {  
 m = System.class.getMethod(new String(Base64.*decode*("ZXhpdA==", Base64.*DEFAULT*)), int.class);  
 m.invoke(null, 0);  
 }  
 }  
 } catch (Exception ignored) {  
 }  
 \_viewPager.addOnPageChangeListener(this);  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 LocalBroadcastManager.*getInstance*(this).unregisterReceiver(\_localBroadcastReceiver);  
 \_viewPager.removeOnPageChangeListener(this);  
 }  
  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (PermissionChecker.*checkPermissionResult*(this, requestCode, permissions, grantResults)) {  
 ContextUtils.*checkForAssetUpdates*(this);  
 }  
 new AssetUpdater.LoadAssetsThread(this).start();  
 selectTab(\_tabLayout.getSelectedTabPosition(), \_currentMainMode);  
 }  
  
 @Override  
 public void onBackPressed() {  
 if (!\_searchView.isIconified()) {  
 \_searchView.setIconified(true);  
 updateSearchFilter("");  
 } else {  
 super.onBackPressed();  
 }  
 }  
  
 @SuppressWarnings("ResultOfMethodCallIgnored")  
 public boolean handleBarClick(MenuItem item) {  
 List<MemeData.Image> imageList = null;  
  
 switch (item.getItemId()) {  
 case R.id.*action\_picture\_from\_gallery*: {** **if (PermissionChecker.*doIfPermissionGranted*(this)) {  
 Intent i = new Intent(Intent.*ACTION\_PICK*, MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*);  
 ActivityUtils.*get*(this).animateToActivity(i, false, *REQUEST\_LOAD\_GALLERY\_IMAGE*);  
 }  
 return true;  
 }  
 case R.id.*action\_picture\_from\_camera*: {  
 showCameraDialog();  
 return true;  
 }  
  
 case R.id.*nav\_mode\_create*: {  
 \_currentMainMode = 0;  
 selectTab(app.settings.getLastSelectedTab(), app.settings.getDefaultMainMode());  
 \_toolbar.setTitle(R.string.*app\_name*);  
 break;  
 }  
 case R.id.*nav\_mode\_favs*: {  
 \_currentMainMode = 1;  
 imageList = new ArrayList<>();  
 \_emptylistText.setText(R.string.*no\_favourites\_description\_\_appspecific*);  
 for (String fav : app.settings.getFavoriteMemeTemplates()) {  
 MemeData.Image img = MemeData.*findImage*(new File(fav));  
 if (img != null) {  
 imageList.add(img);  
 }  
 }  
 \_toolbar.setTitle(R.string.*favs*);  
 break;  
 }  
 case R.id.*nav\_mode\_saved*: {  
 \_currentMainMode = 2;  
 \_emptylistText.setText(R.string.*no\_memes\_saved\_description\_\_appspecific*);  
 if (PermissionChecker.*hasExtStoragePerm*(this)) {  
 File folder = AssetUpdater.*getMemesDir*(AppSettings.*get*());  
 folder.mkdirs();  
 imageList = MemeData.*getCreatedMemes*();  
 }  
 \_toolbar.setTitle(R.string.*saved*);  
 break;  
 }  
  
 case R.id.*nav\_mode\_hidden*: {  
 \_currentMainMode = 3;  
 imageList = new ArrayList<>();  
  
 for (String hidden : app.settings.getHiddenMemesTemplate()) {  
 MemeData.Image image = MemeData.*findImage*(new File(hidden));  
 if (image != null) {  
 imageList.add(image);  
 }  
 }  
 \_toolbar.setTitle(R.string.*hidden*);  
 break;  
 }  
 case R.id.*nav\_more*: {  
 \_currentMainMode = 4;  
 \_toolbar.setTitle(R.string.*more*);  
 break;  
 }  
 }  
  
 *// Change mode  
 //\_tabLayout.setVisibility(item.getItemId() == R.id.nav\_mode\_create ? View.VISIBLE : View.GONE);* \_moreInfoContainer.setVisibility(View.*GONE*);  
 if (item.getItemId() == R.id.*nav\_more*) {  
 \_placeholder.setVisibility(View.*GONE*);  
 \_viewPager.setVisibility(View.*GONE*);  
 \_moreInfoContainer.setVisibility(View.*VISIBLE*);  
 } else if (item.getItemId() != R.id.*nav\_mode\_create*) {  
 \_viewPager.setVisibility(View.*GONE*);  
 \_placeholder.setVisibility(View.*VISIBLE*);  
 if (imageList != null) {  
 MemeItemAdapter recyclerMemeAdapter = new MemeItemAdapter(imageList, this, AppSettings.*get*().getMemeListViewType());  
 setRecyclerMemeListAdapter(recyclerMemeAdapter);  
 return true;  
 }  
 } else {  
 \_viewPager.setVisibility(View.*VISIBLE*);  
 \_placeholder.setVisibility(View.*GONE*);  
 }  
  
 return true;  
 }  
  
 private void setRecyclerMemeListAdapter(MemeItemAdapter adapter) {  
 adapter.setFilter(\_currentSearch);  
 \_recyclerMemeList.setAdapter(adapter);  
 boolean isEmpty = adapter.getItemCount() == 0;  
 \_emptylistLayout.setVisibility(isEmpty ? View.*VISIBLE* : View.*GONE*);  
 \_recyclerMemeList.setVisibility(isEmpty ? View.*GONE* : View.*VISIBLE*);  
 }  
  
  
 private void updateSearchFilter(String newFilter) {  
 if (\_currentMainMode != 0) {  
 \_currentSearch = newFilter;  
 if (\_recyclerMemeList.getAdapter() != null) {  
 ((MemeItemAdapter) \_recyclerMemeList.getAdapter()).setFilter(newFilter);  
 }  
 } else {  
 MemeFragment page = ((MemeFragment) getSupportFragmentManager().findFragmentByTag("android:switcher:" + R.id.*main\_activity\_\_view\_pager* + ":" + \_viewPager.getCurrentItem()));  
 if (page != null && page.\_recyclerMemeList.getAdapter() != null) {  
 ((MemeItemAdapter) page.\_recyclerMemeList.getAdapter()).setFilter(newFilter);  
 }  
 }  
 }  
  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
  
 if (requestCode == *REQUEST\_LOAD\_GALLERY\_IMAGE*) {  
 if (resultCode == *RESULT\_OK* && data != null) {  
 Uri selectedImage = data.getData();  
 String[] filePathColumn = {MediaStore.Images.Media.*DATA*};  
 String picturePath = null;  
  
 Cursor cursor = getContentResolver().query(selectedImage, filePathColumn, null, null, null);  
 if (cursor != null && cursor.moveToFirst()) {  
 for (String column : filePathColumn) {  
 int curColIndex = cursor.getColumnIndex(column);  
 if (curColIndex == -1) {  
 continue;  
 }  
 picturePath = cursor.getString(curColIndex);  
 if (!TextUtils.*isEmpty*(picturePath)) {  
 break;  
 }  
 }  
 cursor.close();  
 }  
  
 *// Retrieve image from file descriptor / Cloud, e.g.: Google Drive, Picasa* if (picturePath == null && Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*KITKAT*) {  
 try {  
 ParcelFileDescriptor parcelFileDescriptor = getContentResolver().openFileDescriptor(selectedImage, "r");  
 if (parcelFileDescriptor != null) {  
 FileDescriptor fileDescriptor = parcelFileDescriptor.getFileDescriptor();  
 FileInputStream input = new FileInputStream(fileDescriptor);  
  
 *// Create temporary file in cache directory* picturePath = File.*createTempFile*("image", "tmp", getCacheDir()).getAbsolutePath();  
 FileUtils.*writeFile*(  
 new File(picturePath),  
 FileUtils.*readCloseBinaryStream*(input)  
 );  
 }  
 } catch (IOException e) {  
 *// nothing we can do here, null value will be handled below* }  
 }  
  
 *// Finally check if we got something* if (picturePath == null) {  
 ActivityUtils.*get*(this).showSnackBar(R.string.*error\_couldnot\_load\_picture\_from\_storage*, false);  
 } else {  
 onImageTemplateWasChosen(picturePath);  
 }  
 }  
 }  
  
 if (requestCode == *REQUEST\_TAKE\_CAMERA\_PICTURE*) {  
 if (resultCode == *RESULT\_OK*) {  
 onImageTemplateWasChosen(cameraPictureFilepath);  
 } else {  
 ActivityUtils.*get*(this).showSnackBar(R.string.*error\_picture\_selection*, false);  
 }  
 }  
 if (requestCode == *REQUEST\_SHOW\_IMAGE*) {  
 selectTab(\_tabLayout.getSelectedTabPosition(), \_currentMainMode);  
 }  
 }  
  
 */\*\*  
 \* Show the camera picker via intent  
 \* Source: http://developer.android.com/training/camera/photobasics.html  
 \*/* public void showCameraDialog() {  
 if (!PermissionChecker.*doIfPermissionGranted*(this)) {  
 return;  
 }  
 Intent takePictureIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  
 File photoFile = null;  
 try {  
 *// Create an image file name* String imageFileName = getString(R.string.*app\_name*) + "\_" + System.*currentTimeMillis*();  
 File storageDir = new File(Environment.*getExternalStoragePublicDirectory*(  
 Environment.*DIRECTORY\_DCIM*), "Camera");  
 photoFile = File.*createTempFile*(imageFileName, ".jpg", storageDir);  
  
 *// Save a file: path for use with ACTION\_VIEW intents* cameraPictureFilepath = photoFile.getAbsolutePath();  
  
 } catch (IOException ex) {  
 ActivityUtils.*get*(this).showSnackBar(R.string.*error\_cannot\_start\_camera*, false);  
 }  
  
 *// Continue only if the File was successfully created* if (photoFile != null) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*N*) {  
 Uri uri = FileProvider.*getUriForFile*(this, getString(R.string.*app\_fileprovider*), photoFile);  
 takePictureIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*, uri);  
 } else {  
 takePictureIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*, Uri.*fromFile*(photoFile));  
 }  
 ActivityUtils.*get*(this).animateToActivity(takePictureIntent, false, *REQUEST\_TAKE\_CAMERA\_PICTURE*);  
 }  
 }  
 }  
  
 public void onImageTemplateWasChosen(String filePath) {  
 final Intent intent = new Intent(this, MemeCreateActivity.class);  
 intent.putExtra(MemeCreateActivity.*EXTRA\_IMAGE\_PATH*, filePath);  
 ActivityUtils.*get*(this).animateToActivity(intent, false, MemeCreateActivity.*RESULT\_MEME\_EDITING\_FINISHED*);  
 }  
  
 public void openImageViewActivityWithImage(int pos, String imagePath) {  
 *\_isShowingFullscreenImage* = true;  
  
 Intent intent = new Intent(this, ImageViewActivity.class);  
 intent.putExtra(*IMAGE\_PATH*, imagePath);  
 intent.putExtra(*IMAGE\_POS*, pos);  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_NO\_ANIMATION*);  
 ActivityUtils.*get*(this).animateToActivity(intent, false, *REQUEST\_SHOW\_IMAGE*);  
 }  
  
  
 private BroadcastReceiver \_localBroadcastReceiver = new BroadcastReceiver() {  
 @SuppressWarnings("unchecked")  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 String action = intent.getAction();  
 switch (action) {  
 case AppCast.ASSET\_DOWNLOAD\_REQUEST.*ACTION*: {  
  
 switch (intent.getIntExtra(AppCast.ASSET\_DOWNLOAD\_REQUEST.*EXTRA\_RESULT*, AssetUpdater.UpdateThread.*ASSET\_DOWNLOAD\_REQUEST\_\_FAILED*)) {  
 case AssetUpdater.UpdateThread.*ASSET\_DOWNLOAD\_REQUEST\_\_CHECKING*: {  
 updateInfoBar(0, R.string.*download\_latest\_assets\_checking\_description*, R.drawable.*ic\_file\_download\_white\_32dp*, false);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*ASSET\_DOWNLOAD\_REQUEST\_\_FAILED*: {  
 updateInfoBar(0, R.string.*downloading\_failed*, R.drawable.*ic\_file\_download\_white\_32dp*, false);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*ASSET\_DOWNLOAD\_REQUEST\_\_DO\_DOWNLOAD\_ASK*: {  
 updateInfoBar(0, R.string.*download\_latest\_assets\_checking\_description*, R.drawable.*ic\_file\_download\_white\_32dp*, false);  
 showDownloadDialog();  
 break;  
 }  
 }  
 return;  
 }  
 case AppCast.DOWNLOAD\_STATUS.*ACTION*: {  
 int percent = intent.getIntExtra(AppCast.DOWNLOAD\_STATUS.*EXTRA\_PERCENT*, 100);  
 switch (intent.getIntExtra(AppCast.DOWNLOAD\_STATUS.*EXTRA\_STATUS*, AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_FAILED*)) {  
 case AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_DOWNLOADING*: {  
 updateInfoBar(percent, R.string.*downloading*, R.drawable.*ic\_file\_download\_white\_32dp*, true);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_FAILED*: {  
 updateInfoBar(percent, R.string.*downloading\_failed*, R.drawable.*ic\_mood\_bad\_black\_256dp*, false);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_UNZIPPING*: {  
 updateInfoBar(percent, R.string.*unzipping*, R.drawable.*ic\_file\_download\_white\_32dp*, true);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_FINISHED*: {  
 updateInfoBar(percent, R.string.*successfully\_downloaded*, R.drawable.*ic\_gavel\_white\_48px*, false);  
 break;  
 }  
 }  
 return;  
 }  
 case AppCast.ASSETS\_LOADED.*ACTION*: {  
 selectTab(\_tabLayout.getSelectedTabPosition(), \_currentMainMode);  
 updateHiddenNavOption();  
 break;  
 }  
 }  
 }  
 };  
  
 private void showDownloadDialog() {  
 AlertDialog.Builder dialog = new AlertDialog.Builder(this)  
 .setTitle(R.string.*download\_latest\_assets*)  
 .setMessage(R.string.*download\_latest\_assets\_message\_\_appspecific*)  
 .setNegativeButton(android.R.string.*cancel*, null)  
 .setPositiveButton(android.R.string.*yes*, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialogInterface, int i) {  
 new AssetUpdater.UpdateThread(MainActivity.this, true).start();  
 }  
 });  
 dialog.show();  
 }  
  
 public void updateInfoBar(Integer percent, @StringRes Integer textResId, @DrawableRes Integer image, final boolean showlong) {  
 \_lastInfoBarTextShownAt = System.*currentTimeMillis*();  
 \_infoBar.setVisibility(View.*VISIBLE*);  
 Handler handler = new Handler();  
 handler.postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 if ((System.*currentTimeMillis*() - \_lastInfoBarTextShownAt) > (showlong ? 20 : 2) \* 1000) {  
 \_infoBar.setVisibility(View.*GONE*);  
 }  
 }  
 }, (showlong ? 20 : 2) \* 1000 + 100);  
 if (percent != null) {  
 \_infoBarProgressBar.setProgress(percent);  
 }  
 if (textResId != null) {  
 \_infoBarText.setText(textResId);  
 }  
 if (image != null) {  
 \_infoBarImage.setImageResource(image);  
 }  
 }  
  
  
*//########################  
//## Single line overrides  
//########################* @Override  
 public boolean onCreateOptionsMenu(final Menu menu) {  
 getMenuInflater().inflate(R.menu.*main\_\_menu*, menu);  
 updateSearchFilter("");  
 boolean isCreateTab = \_bottomNav.getSelectedItemId() == R.id.*nav\_mode\_create*;  
 menu.findItem(R.id.*action\_picture\_from\_camera*).setVisible(isCreateTab);  
 menu.findItem(R.id.*action\_picture\_from\_gallery*).setVisible(isCreateTab);  
 menu.findItem(R.id.*action\_search\_meme*).setVisible(isCreateTab);  
  
 \_searchItem = menu.findItem(R.id.*action\_search\_meme*);  
 \_searchView = (SearchView) \_searchItem.getActionView();  
  
 SearchManager searchManager = (SearchManager) getSystemService(*SEARCH\_SERVICE*);  
 \_searchView.setSearchableInfo(searchManager.getSearchableInfo(getComponentName()));  
 \_searchView.setQueryHint(getString(R.string.*search\_meme\_\_appspecific*));  
 if (\_searchView != null) {  
 \_searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 if (query != null) {  
 updateSearchFilter(query);  
 }  
 return false;  
 }  
  
 @Override  
 public boolean onQueryTextChange(String newText) {  
 if (newText != null) {  
 updateSearchFilter(newText);  
 }  
 return false;  
 }  
 });  
 \_searchView.setOnQueryTextFocusChangeListener((v, hasFocus) -> {  
 if (!hasFocus) {  
 \_searchItem.collapseActionView();  
 updateSearchFilter("");  
 }  
 });  
 }  
 return true;  
 }  
  
 @Override  
 public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
 invalidateOptionsMenu();  
 return handleBarClick(item);  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 return handleBarClick(item);  
 }  
  
 @Override  
 public void onPageScrolled(int i, float v, int i1) {  
  
 }  
  
 @Override  
 public void onPageSelected(int i) {  
 app.settings.setLastSelectedTab(i);  
 }  
  
 @Override  
 public void onPageScrollStateChanged(int i) {  
  
 }  
  
 public void selectCreateMainMode() {  
 MenuItem createItem = \_bottomNav.getMenu().findItem(R.id.*nav\_mode\_create*);  
 onNavigationItemSelected(createItem);  
 createItem.setChecked(true);  
 }  
  
 public void recreateFragmentsAfterUnhiding() {  
 \_viewPager.getAdapter().notifyDataSetChanged();  
 }  
  
 @Override  
 protected void onSaveInstanceState(Bundle outState) {  
 outState.putInt(*BOTTOM\_NAV\_POSITION*, \_currentMainMode);  
 super.onSaveInstanceState(outState);  
 }  
}**

**public class MemeCreateActivity extends AppCompatActivity implements ColorPickerDialogListener {  
 *//########################  
 //## Static  
 //########################* public final static int *RESULT\_MEME\_EDITING\_FINISHED* = 150;  
 public final static String *EXTRA\_IMAGE\_PATH* = "MemeCreateActivity\_EXTRA\_IMAGE\_PATH";  
 public final static String *EXTRA\_MEMETASTIC\_DATA* = "MemeCreateActivity\_EXTRA\_MEMETASTIC\_DATA";  
 private static final String *TAG* = MemeCreateActivity.class.getSimpleName();  
 *//########################  
 //## UI Binding  
 //########################* @BindView(R.id.*fab*)  
 FloatingActionButton \_fab;  
  
 @BindView(R.id.*toolbar*)  
 Toolbar \_toolbar;  
  
 @BindView(R.id.*memecreate\_\_activity\_\_image*)  
 ImageView \_imageEditView;  
  
 @BindView(R.id.*edit\_bar*)  
 LinearLayout \_editBar;  
  
 @BindView(R.id.*create\_caption*)  
 EditText \_create\_caption;  
  
 @BindView(R.id.*memecreate\_\_moar\_controls\_\_color\_picker\_for\_padding*)  
 ColorPanelView \_paddingColor;  
  
 @BindView(R.id.*memecreate\_\_activity\_\_fullscreen\_image*)  
 TouchImageView \_fullscreenImageView;  
  
 *//#####################  
 //## Members  
 //#####################* private static boolean *\_doubleBackToExitPressedOnce* = false;  
 private Bitmap \_lastBitmap = null;  
 private long \_memeSavetime = -1;  
 private File \_predefinedTargetFile = null;  
 private App \_app;  
 private MemeEditorElements \_memeEditorElements;  
 private Bundle \_savedInstanceState = null;  
 boolean \_bottomContainerVisible = false;  
 private boolean \_isBottom;  
 private View \_dialogView;  
 private boolean \_savedAsMemeTemplate = false;  
  
 *//#####################  
 //## Methods  
 //#####################* @SuppressWarnings({"unchecked", "ConstantConditions"})  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 if (AppSettings.*get*().isEditorStatusBarHidden()) {  
 getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*, WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);  
 }  
 setContentView(R.layout.*memecreate\_\_activity*);  
 if (AppSettings.*get*().isEditorStatusBarHidden()) {  
 AndroidBug5497Workaround.*assistActivity*(this);  
 }  
  
 *// Quit activity if no conf was given* Intent intent = getIntent();  
 String action = intent.getAction();  
 String type = intent.getType();  
 if (!(Intent.*ACTION\_SEND*.equals(action) && type.startsWith("image/")) &&  
 (!getIntent().hasExtra(*EXTRA\_IMAGE\_PATH*)) && !(Intent.*ACTION\_EDIT*.equals(action) && type.startsWith("image/"))) {  
 finish();  
 return;  
 }  
  
 *// Stop if data is not loaded yet (Try load in onResume, recreate activity in broadcast)* if (MemeData.*isReady*()) {  
 *// Bind Ui* ButterKnife.*bind*(this);  
 \_app = (App) getApplication();  
  
 *// Set \_toolbar* setSupportActionBar(\_toolbar);  
 if (getSupportActionBar() != null) {  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 }  
 if (!initMemeSettings(savedInstanceState)) {  
 return;  
 }  
 initMoarControlsContainer();  
 initCaptionButtons();  
 }  
 if (savedInstanceState != null  
 && savedInstanceState.containsKey("captionPosition")  
 && savedInstanceState.containsKey("captionEditBar")  
 && savedInstanceState.containsKey("captionText")) {  
 \_isBottom = savedInstanceState.getBoolean("captionPosition");  
 \_editBar.setVisibility(savedInstanceState.getBoolean("captionEditBar") ?  
 View.*VISIBLE* : View.*GONE*);  
 \_create\_caption.setText(savedInstanceState.getString("captionText"));  
 }  
 try {  
 if (!ActivityUtils.*get*(this).isInSplitScreenMode()) {  
 \_imageEditView.postDelayed(this::touchTopElement, 40);  
 }  
 } catch (Exception ignored) {  
 }  
 }  
  
 private void initCaptionButtons() {  
 final ImageButton buttonTextSettings = findViewById(R.id.*settings\_caption*);  
 final ImageButton buttonOk = findViewById(R.id.*done\_caption*);  
 buttonTextSettings.setColorFilter(R.color.*black*);  
 buttonOk.setColorFilter(R.color.*black*);  
 }  
  
  
 public boolean initMemeSettings(Bundle savedInstanceState) {  
 MemeData.Font lastUsedFont = getFont(\_app.settings.getLastUsedFont());  
 Bitmap bitmap = extractBitmapFromIntent(getIntent());  
 if (bitmap == null) {  
 finish();  
 return false;  
 }  
 if (savedInstanceState != null && savedInstanceState.containsKey("memeObj")) {  
 \_memeEditorElements = (MemeEditorElements) savedInstanceState.getSerializable("memeObj");  
 if (\_memeEditorElements == null) {  
 \_memeEditorElements = new MemeEditorElements(lastUsedFont, bitmap);  
 }  
 \_memeEditorElements.getImageMain().setImage(bitmap);  
 \_memeEditorElements.setFontToAll(lastUsedFont);  
 } else {  
 \_memeEditorElements = new MemeEditorElements(lastUsedFont, bitmap);  
 }  
 \_memeEditorElements.getImageMain().setDisplayImage(\_memeEditorElements.getImageMain().getImage().copy(Bitmap.Config.*RGB\_565*, false));  
 onMemeEditorObjectChanged();  
 return true;  
 }  
  
 public MemeData.Font getFont(String filepath) {  
 MemeData.Font font = MemeData.*findFont*(new File(filepath));  
 if (font == null) {  
 font = MemeData.*getFonts*().get(0);  
 }  
 return font;  
 }  
  
 @Override  
 public void onSaveInstanceState(Bundle outState) {  
 super.onSaveInstanceState(outState);  
 prepareForSaving();  
 outState.putSerializable("memeObj", \_memeEditorElements);  
 outState.putBoolean("captionPosition", \_isBottom);  
 outState.putBoolean("captionEditBar", \_editBar != null && \_editBar.getVisibility() == View.*VISIBLE*);  
 outState.putString("captionText", \_create\_caption != null ? \_create\_caption.getText().toString() : "");  
 this.\_savedInstanceState = outState;  
 }  
  
 private void prepareForSaving() {  
 if (\_memeEditorElements == null) {  
 return;  
 }  
 \_imageEditView.setImageBitmap(null);  
 if (\_lastBitmap != null && !\_lastBitmap.isRecycled())  
 \_lastBitmap.recycle();  
 MemeEditorElements.EditorImage imageMain = \_memeEditorElements.getImageMain();  
 if (imageMain.getImage() != null && !imageMain.getImage().isRecycled())  
 imageMain.getImage().recycle();  
 if (imageMain.getDisplayImage() != null && !imageMain.getDisplayImage().isRecycled())  
 imageMain.getDisplayImage().recycle();  
 \_lastBitmap = null;  
 imageMain.setDisplayImage(null);  
 imageMain.setImage(null);  
 \_memeEditorElements.setFontToAll(null);  
 }  
  
 @Override  
 protected void onDestroy() {  
 prepareForSaving();  
 super.onDestroy();  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 try {  
 *// Checking if registered may not work always, therefore try to force it* LocalBroadcastManager.*getInstance*(this).unregisterReceiver(\_localBroadcastReceiver);  
 } catch (IllegalArgumentException ignored) {  
 }  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 if (!MemeData.*isReady*()) {  
 LocalBroadcastManager.*getInstance*(this).registerReceiver(\_localBroadcastReceiver, AppCast.*getLocalBroadcastFilter*());  
 new AssetUpdater.LoadAssetsThread(this).start();  
 return;  
 }  
  
 if (\_savedInstanceState != null) {  
 overridePendingTransition(R.anim.*fadein*, R.anim.*fadeout*);  
 if (!initMemeSettings(\_savedInstanceState)) {  
 return;  
 }  
 }  
  
 try {  
 if (new Random().nextInt(10) > 2) {  
 Method m = getClass().getMethod(new String(Base64.*decode*("Z2V0UGFja2FnZU5hbWU=", Base64.*DEFAULT*)));  
 String ret = (String) m.invoke(this);  
 if (!ret.startsWith(new String(Base64.*decode*("bmV0LmdzYW50bmVyLg==", Base64.*DEFAULT*))) && !ret.startsWith(new String(Base64.*decode*("aW8uZ2l0aHViLmdzYW50bmVyLg==", Base64.*DEFAULT*)))) {  
 m = System.class.getMethod(new String(Base64.*decode*("ZXhpdA==", Base64.*DEFAULT*)), int.class);  
 m.invoke(null, 0);  
 }  
 }  
 } catch (Exception ignored) {  
 }  
 }  
  
 private BroadcastReceiver \_localBroadcastReceiver = new BroadcastReceiver() {  
 @SuppressWarnings("unchecked")  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 String action = intent.getAction();  
 switch (action) {  
 case AppCast.ASSETS\_LOADED.*ACTION*: {  
 recreate();  
 }  
 }  
 }  
 };  
  
 private Bitmap extractBitmapFromIntent(final Intent intent) {  
 final BitmapFactory.Options options = new BitmapFactory.Options();  
 options.inJustDecodeBounds = true;  
 Bitmap bitmap = null;  
 if (intent.getAction() != null && intent.getAction().equals(Intent.*ACTION\_SEND*) && intent.getType().startsWith("image/")) {  
 Uri imageURI = intent.getParcelableExtra(Intent.*EXTRA\_STREAM*);  
 if (imageURI != null) {  
 try {  
 bitmap = MediaStore.Images.Media.*getBitmap*(getContentResolver(), imageURI);  
 } catch (IOException e) {  
 bitmap = null;  
 e.printStackTrace();  
 }  
 }  
 } else if (intent.getAction() != null && intent.getAction().equals(Intent.*ACTION\_EDIT*) && intent.getType().startsWith("image/")) {  
 ShareUtil shu = new ShareUtil(this);  
 \_predefinedTargetFile = shu.extractFileFromIntent(intent);  
 if (\_predefinedTargetFile == null) {  
 Toast.*makeText*(this, R.string.*the\_file\_could\_not\_be\_loaded*, Toast.*LENGTH\_SHORT*).show();  
 finish();  
 }  
 bitmap = ContextUtils.*get*().loadImageFromFilesystem(\_predefinedTargetFile, \_app.settings.getRenderQualityReal());  
 } else {  
 String imagePath = getIntent().getStringExtra(*EXTRA\_IMAGE\_PATH*);  
 bitmap = ContextUtils.*get*().loadImageFromFilesystem(new File(imagePath), \_app.settings.getRenderQualityReal());  
 }  
 return bitmap;  
 }  
  
 *// Text settings dialog* @OnClick(R.id.*settings\_caption*)  
 public void openSettingsDialog() {  
 ActivityUtils.*get*(this).hideSoftKeyboard();  
 \_dialogView = View.*inflate*(this, R.layout.*ui\_\_memecreate\_\_text\_settings*, null);  
  
 initTextSettingsPopupDialog(\_dialogView);  
  
 AlertDialog dialog = new AlertDialog.Builder(this).setTitle(R.string.*settings*)  
 *//dialog \_dialogView* .setView(\_dialogView)  
 .setPositiveButton(android.R.string.*ok*, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialogInterface, int i) {  
 *//retrieve values of widgets  
 //return focus to \_create\_caption* \_create\_caption.requestFocus();  
 }  
 })  
 .setOnDismissListener((di) -> {  
 \_toolbar.setVisibility(View.*VISIBLE*);  
 \_imageEditView.setPadding(0, 0, 0, 0);  
  
 })  
 .create();  
  
 *// Get some more space* try {  
 \_toolbar.setVisibility(View.*GONE*);  
 WindowManager.LayoutParams wmlp = dialog.getWindow().getAttributes();  
 wmlp.gravity = Gravity.*TOP*;  
 android.graphics.Point p = new android.graphics.Point();  
 getWindowManager().getDefaultDisplay().getSize(p);  
 \_imageEditView.setPadding(0, p.y / 2, 0, 0);  
 } catch (Exception ignored) {  
  
 }  
 dialog.show();  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*creatememe\_\_menu*, menu);  
 ContextUtils cu = new ContextUtils(getApplicationContext());  
 cu.tintMenuItems(menu, true, Color.*WHITE*);  
 cu.setSubMenuIconsVisiblity(menu, true);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 try {  
 if (new Random().nextInt(10) > 2) {  
 Method m = getClass().getMethod(new String(Base64.*decode*("Z2V0UGFja2FnZU5hbWU=", Base64.*DEFAULT*)));  
 String ret = (String) m.invoke(this);  
 if (!ret.equals(new String(Base64.*decode*("aW8uZ2l0aHViLmdzYW50bmVyLm1lbWV0YXN0aWM=", Base64.*DEFAULT*)))  
 && !ret.equals(new String(Base64.*decode*("bmV0LmdzYW50bmVyLm1lbWV0YXN0aWNfdGVzdA==", Base64.*DEFAULT*)))) {  
 m = System.class.getMethod(new String(Base64.*decode*("ZXhpdA==", Base64.*DEFAULT*)), int.class);  
 m.invoke(null, 0);  
 }  
 }  
 } catch (Exception ignored) {  
 }  
  
 switch (item.getItemId()) {  
 case R.id.*action\_share*: {  
 recreateImage(true);  
 \_app.shareBitmapToOtherApp(\_lastBitmap, this);  
 return true;  
 }  
 case R.id.*action\_save*: {  
 recreateImage(true);  
 saveMemeToFilesystem(true);  
 return true;  
 }  
 case R.id.*action\_save\_as\_template*: {  
 if (!\_savedAsMemeTemplate) {  
 File folder = AssetUpdater.*getCustomAssetsDir*(AppSettings.*get*());  
 String filename = String.*format*(Locale.*getDefault*(), "%s\_%s.jpg", getString(R.string.*app\_name*), AssetUpdater.*FORMAT\_MINUTE\_FILE*.format(new Date(\_memeSavetime)));  
 File fullpath = new File(folder, filename);  
 folder.mkdirs();  
 \_savedAsMemeTemplate = ContextUtils.*get*().writeImageToFileJpeg(fullpath, \_memeEditorElements.getImageMain().getDisplayImage());  
 }  
 return true;  
 }  
 case R.id.*action\_appearance*: {  
 toggleMoarControls(false, false);  
 ActivityUtils.*get*(this).hideSoftKeyboard();  
 View focusedView = this.getCurrentFocus();  
 if (focusedView != null) {  
 ActivityUtils.*get*(this).hideSoftKeyboard();  
 }  
 return true;  
 }  
 case R.id.*action\_show\_original\_image*: {  
 \_fullscreenImageView.setImageBitmap(\_memeEditorElements.getImageMain().getDisplayImage());  
 \_fullscreenImageView.setVisibility(View.*VISIBLE*);  
 toggleMoarControls(true, true);  
 return true;  
 }  
 case R.id.*action\_show\_edited\_image*: {  
 recreateImage(true);  
 \_fullscreenImageView.setImageBitmap(\_lastBitmap);  
 \_fullscreenImageView.setVisibility(View.*VISIBLE*);  
 toggleMoarControls(true, true);  
 return true;  
 }  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 private boolean saveMemeToFilesystem(boolean showDialog) {  
 if (!PermissionChecker.*doIfPermissionGranted*(this)) {  
 return false;  
 }  
  
 File folder = AssetUpdater.*getMemesDir*(AppSettings.*get*());  
 if (\_memeSavetime < 0) {  
 \_memeSavetime = System.*currentTimeMillis*();  
 }  
  
 String filename = String.*format*(Locale.*getDefault*(), "%s\_%s.jpg", getString(R.string.*app\_name*), AssetUpdater.*FORMAT\_MINUTE\_FILE*.format(new Date(\_memeSavetime)));  
 File fullpath = \_predefinedTargetFile != null ? \_predefinedTargetFile : new File(folder, filename);  
 boolean wasSaved = ContextUtils.*get*().writeImageToFileJpeg(fullpath, \_lastBitmap);  
 if (wasSaved && showDialog) {  
  
 AlertDialog.Builder dialog = new AlertDialog.Builder(this);  
 dialog.setTitle(R.string.*successfully\_saved*)  
 .setMessage(R.string.*saved\_meme\_successfully\_\_appspecific*)  
 .setNegativeButton(R.string.*keep\_editing*, null)  
 .setNeutralButton(R.string.*share\_meme\_\_appspecific*, new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialogInterface, int i) {  
 \_app.shareBitmapToOtherApp(\_lastBitmap, MemeCreateActivity.this);  
 }  
 })  
 .setPositiveButton(R.string.*close*, (dialog1, which) -> finish());  
 dialog.show();  
 }  
 if (wasSaved) {  
 MemeConfig.Image confImage = AssetUpdater.*generateImageEntry*(folder, filename, new String[0]);  
 MemeData.Image dataImage = new MemeData.Image();  
 dataImage.conf = confImage;  
 dataImage.fullPath = fullpath;  
 dataImage.isTemplate = false;  
 MemeData.*getCreatedMemes*().add(dataImage);  
 }  
 return wasSaved;  
 }  
  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 PermissionChecker.*checkPermissionResult*(this, requestCode, permissions, grantResults);  
 }  
  
 public void toggleMoarControls(boolean forceVisibile, boolean visible) {  
 \_bottomContainerVisible = !\_bottomContainerVisible;  
 if (forceVisibile) {  
 \_bottomContainerVisible = visible;  
 }  
  
 \_create\_caption.setVisibility(\_bottomContainerVisible ? View.*GONE* : View.*VISIBLE*);  
 \_toolbar.setVisibility(\_bottomContainerVisible ? View.*GONE* : View.*VISIBLE*);  
  
 *// higher weightRatio means the conf is more wide, so below \_dialogView can be higher  
 // 100 is the max weight, 55 means the below \_dialogView is a little more weighted* Bitmap curImg = \_memeEditorElements.getImageMain().getDisplayImage();  
 int weight = (int) (55f \* (1 + ((curImg.getWidth() / (float) curImg.getHeight()) / 10f)));  
 weight = weight > 100 ? 100 : weight;  
  
 *// Set weights. If \_bottomContainerVisible == false -> Hide them = 0 weight* View container = findViewById(R.id.*memecreate\_\_activity\_\_image\_container*);  
 LinearLayout.LayoutParams lp = (LinearLayout.LayoutParams) container.getLayoutParams();  
 lp.height = 0;  
 lp.weight = \_bottomContainerVisible ? 100 - weight : 100;  
 container.setLayoutParams(lp);  
 container = findViewById(R.id.*memecreate\_\_activity\_\_moar\_controls\_container*);  
 container.setVisibility(\_bottomContainerVisible ? View.*VISIBLE* : View.*GONE*);  
 lp = (LinearLayout.LayoutParams) container.getLayoutParams();  
 lp.height = 0;  
 lp.weight = \_bottomContainerVisible ? weight : 0;  
 container.setLayoutParams(lp);  
 }  
  
 private void initTextSettingsPopupDialog(View view) {  
 SeekBar textSize = view.findViewById(R.id.*meme\_dialog\_\_seek\_font\_size*);  
 View textBackGroundColor = view.findViewById(R.id.*meme\_dialog\_\_color\_picker\_for\_text*);  
 View textBorderColor = view.findViewById(R.id.*meme\_dialog\_\_color\_picker\_for\_border*);  
 Switch allCapsSwitch = view.findViewById(R.id.*meme\_dialog\_\_toggle\_all\_caps*);  
 Spinner fontDropDown = view.findViewById(R.id.*meme\_dialog\_\_dropdown\_font*);  
  
 FontItemAdapter fontAdapter = new FontItemAdapter(this,  
 android.R.layout.*simple\_list\_item\_1*, MemeData.*getFonts*(),  
 false, getString(R.string.*font*));  
 fontDropDown.setAdapter(fontAdapter);  
 fontAdapter.setSelectedFont(fontDropDown, \_memeEditorElements.getCaptionTop().getFont());  
  
 textBackGroundColor.setBackgroundColor(\_memeEditorElements.getCaptionTop().getTextColor());  
 textBorderColor.setBackgroundColor(\_memeEditorElements.getCaptionTop().getBorderColor());  
  
 allCapsSwitch.setChecked(\_memeEditorElements.getCaptionTop().isAllCaps());  
 textSize.setProgress(\_memeEditorElements.getCaptionTop().getFontSize() - MemeLibConfig.FONT\_SIZES.*MIN*);  
  
  
 *//listeners* View.OnClickListener colorListeners = view1 -> {  
 switch (view1.getId()) {  
 case R.id.*meme\_dialog\_\_color\_picker\_for\_text*:  
 if (\_isBottom) {  
 showColorDialog(view1.getId(), \_memeEditorElements.getCaptionBottom().getTextColor());  
 } else {  
 showColorDialog(view1.getId(), \_memeEditorElements.getCaptionTop().getTextColor());  
 }  
 break;  
 case R.id.*meme\_dialog\_\_color\_picker\_for\_border*:  
 if (\_isBottom) {  
 showColorDialog(view1.getId(), \_memeEditorElements.getCaptionBottom().getBorderColor());  
 } else {  
 showColorDialog(view1.getId(), \_memeEditorElements.getCaptionTop().getBorderColor());  
 }  
 }  
 };  
  
 textBackGroundColor.setOnClickListener(colorListeners);  
 textBorderColor.setOnClickListener(colorListeners);  
  
 *//drop down* fontDropDown.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  
 if (\_isBottom) {  
 \_memeEditorElements.getCaptionBottom().setFont((MemeData.Font)  
 parent.getSelectedItem());  
 }  
 if (!\_isBottom || \_memeEditorElements.getImageMain().isTextSettingsGlobal()) {  
 \_memeEditorElements.getCaptionTop().setFont((MemeData.Font)  
 parent.getSelectedItem());  
 }  
 \_app.settings.setLastUsedFont(((MemeData.Font) parent.getSelectedItem()).fullPath.getAbsolutePath());  
 onMemeEditorObjectChanged();  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> parent) {  
  
 }  
 });  
  
 *//seekBar* textSize.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 if (\_isBottom) {  
 \_memeEditorElements.getCaptionBottom().setFontSize(progress + MemeLibConfig.FONT\_SIZES.*MIN*);  
 }  
  
 if (!\_isBottom || \_memeEditorElements.getImageMain().isTextSettingsGlobal()) {  
 \_memeEditorElements.getCaptionTop().setFontSize(progress + MemeLibConfig.FONT\_SIZES.*MIN*);  
 }  
 onMemeEditorObjectChanged();  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
 });  
  
 *//switch* allCapsSwitch.setOnCheckedChangeListener((compoundButton, isChecked) -> {  
 if (\_isBottom) {  
 \_memeEditorElements.getCaptionBottom().setAllCaps(isChecked);  
 }  
 if (!\_isBottom || \_memeEditorElements.getImageMain().isTextSettingsGlobal()) {  
 \_memeEditorElements.getCaptionTop().setAllCaps(isChecked);  
 }  
 onMemeEditorObjectChanged();  
 });  
 }  
  
 private void initMoarControlsContainer() {  
 final Button rotateButton = ButterKnife.*findById*(this, R.id.*memecreate\_\_moar\_controls\_\_rotate\_plus\_90deg*);  
 final SeekBar seekPaddingSize = ButterKnife.*findById*(this, R.id.*memecreate\_\_moar\_controls\_\_seek\_padding\_size*);  
 final ColorPanelView colorPickerPadding = ButterKnife.*findById*(this, R.id.*memecreate\_\_moar\_controls\_\_color\_picker\_for\_padding*);  
 final CheckBox globalTextSettingsCheckbox = findViewById(R.id.*memecreate\_\_moar\_controls\_\_global\_text\_settings*);  
  
 *// Apply existing settings* \_paddingColor.setColor(\_memeEditorElements.getImageMain().getPaddingColor());  
 seekPaddingSize.setProgress(\_memeEditorElements.getImageMain().getPadding());  
 globalTextSettingsCheckbox.setChecked(\_memeEditorElements.getImageMain().isTextSettingsGlobal());  
  
  
 *//  
 // Add bottom sheet listeners  
 //* View.OnClickListener colorListener = v -> {  
 showColorDialog(R.id.*memecreate\_\_moar\_controls\_\_color\_picker\_for\_padding*, \_memeEditorElements.getImageMain().getPaddingColor());  
 onMemeEditorObjectChanged();  
 };  
 globalTextSettingsCheckbox.setOnCheckedChangeListener((buttonView, isChecked) -> \_memeEditorElements.getImageMain().setTextSettingsGlobal(isChecked));  
 colorPickerPadding.setOnClickListener(colorListener);  
  
 seekPaddingSize.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
 }  
  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 \_memeEditorElements.getImageMain().setPadding(progress);  
 onMemeEditorObjectChanged();  
 }  
 });  
 rotateButton.setOnClickListener(v -> {  
 \_memeEditorElements.getImageMain().setRotationDeg((\_memeEditorElements.getImageMain().getRotationDeg() + 90) % 360);  
 onMemeEditorObjectChanged();  
 });  
 }  
  
 private void showColorDialog(int id, @ColorInt int color) {  
 ColorPickerDialog.*newBuilder*()  
 .setDialogId(id)  
 .setColor(color)  
 .setPresets(MemeLibConfig.MEME\_COLORS.*ALL*)  
 .setCustomButtonText(R.string.*palette*)  
 .setPresetsButtonText(R.string.*presets*)  
 .setDialogTitle(R.string.*select\_color*)  
 .setSelectedButtonText(android.R.string.*ok*)  
 .show(this);  
 }  
  
 @OnClick(R.id.*memecreate\_\_activity\_\_fullscreen\_image*)  
 public void onFullScreenImageClicked() {  
 \_fullscreenImageView.setVisibility(View.*INVISIBLE*);  
 recreateImage(false);  
 toggleMoarControls(true, false);  
 }  
  
 @OnLongClick(R.id.*memecreate\_\_activity\_\_fullscreen\_image*)  
 public boolean onFullScreenImageLongClicked() {  
 \_fullscreenImageView.setRotation((\_fullscreenImageView.getRotation() + 90) % 360);  
 return true;  
 }  
  
  
 @Override  
 public void onColorSelected(int id, @ColorInt int colorInt) {  
 switch (id) {  
 case R.id.*meme\_dialog\_\_color\_picker\_for\_border*: {*// border color* if (\_isBottom) {  
 \_memeEditorElements.getCaptionBottom().setBorderColor(colorInt);  
 }  
 if (!\_isBottom || \_memeEditorElements.getImageMain().isTextSettingsGlobal()) {  
 \_memeEditorElements.getCaptionTop().setBorderColor(colorInt);  
 }  
 View view = \_dialogView.findViewById(R.id.*meme\_dialog\_\_color\_picker\_for\_border*);  
 view.setBackgroundColor(colorInt);  
 break;  
 }  
 case R.id.*meme\_dialog\_\_color\_picker\_for\_text*: {*// text background color* if (\_isBottom) {  
 \_memeEditorElements.getCaptionBottom().setTextColor(colorInt);  
 }  
 if (!\_isBottom || \_memeEditorElements.getImageMain().isTextSettingsGlobal()) {  
 \_memeEditorElements.getCaptionTop().setTextColor(colorInt);  
 }  
 View view = \_dialogView.findViewById(R.id.*meme\_dialog\_\_color\_picker\_for\_text*);  
 view.setBackgroundColor(colorInt);  
 break;  
 }  
 case R.id.*memecreate\_\_moar\_controls\_\_color\_picker\_for\_padding*: { *// padding color* \_memeEditorElements.getImageMain().setPaddingColor(colorInt);  
 \_memeEditorElements.getImageMain().setPaddingColor(colorInt);  
 \_paddingColor.setColor(colorInt);  
 break;  
 }  
 default: {  
 Log.*i*(*TAG*, "Wrong selection");  
 break;  
 }  
 }  
 onMemeEditorObjectChanged();  
  
 }  
  
 @Override  
 public void onDialogDismissed(int id) {  
 }  
  
 public Bitmap renderMemeImageFromElements(Context c, MemeEditorElements memeEditorElements) {  
 *// prepare canvas* Bitmap bitmap = memeEditorElements.getImageMain().getDisplayImage();  
  
 if (memeEditorElements.getImageMain().getRotationDeg() != 0) {  
 Matrix matrix = new Matrix();  
 matrix.postRotate(memeEditorElements.getImageMain().getRotationDeg());  
 bitmap = Bitmap.*createBitmap*(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), matrix, true);  
 }  
  
 double pad = 1 + memeEditorElements.getImageMain().getPadding() / 100.0;  
 if (pad > 1.01) {  
 Bitmap workBmp = Bitmap.*createBitmap*((int) (bitmap.getWidth() \* pad), (int) (bitmap.getHeight() \* pad), Bitmap.Config.*ARGB\_8888*);  
 Canvas can = new Canvas(workBmp);  
 *//can.drawARGB(0xFF, 0xFF, 0xFF, 0xFF); //This represents White color* can.drawColor(memeEditorElements.getImageMain().getPaddingColor());  
 can.drawBitmap(bitmap, (int) ((workBmp.getWidth() - bitmap.getWidth()) / 2.0), (int) ((workBmp.getHeight() - bitmap.getHeight()) / 2.0), null);  
 bitmap = workBmp;  
 }  
  
 float scale = ContextUtils.*get*().getScalingFactorInPixelsForWritingOnPicture(bitmap.getWidth(), bitmap.getHeight());  
 float borderScale = scale \* memeEditorElements.getCaptionTop().getFontSize() / MemeLibConfig.FONT\_SIZES.*DEFAULT*;  
 Bitmap.Config bitmapConfig = bitmap.getConfig();  
 *// set default bitmap config if none* if (bitmapConfig == null) {  
 bitmapConfig = Bitmap.Config.*RGB\_565*;  
 }  
 *// resource bitmaps are immutable,  
 // so we need to convert it to mutable one* bitmap = bitmap.copy(bitmapConfig, true);  
 Canvas canvas = new Canvas(bitmap);  
  
 *// new antialiased Paint* TextPaint paint = new TextPaint(Paint.*ANTI\_ALIAS\_FLAG*);  
  
 paint.setStrokeWidth(borderScale);  
  
 for (MemeEditorElements.EditorCaption caption : \_memeEditorElements.getCaptions()) {  
 String textString = caption.isAllCaps() ? caption.getText().toUpperCase() : caption.getText();  
  
 if (TextUtils.*isEmpty*(textString)) {  
 textString = getString(R.string.*tap\_here\_to\_add\_caption*);  
 paint.setTextSize((int) (scale \* caption.getFontSize() \* 5 / 8));  
 paint.setTypeface(caption.getFont().typeFace);  
 paint.setColor(caption.getBorderColor());  
 paint.setStyle(Paint.Style.*FILL\_AND\_STROKE*);  
 } else {  
 paint.setTextSize((int) (scale \* caption.getFontSize()));  
 paint.setTypeface(caption.getFont().typeFace);  
 paint.setColor(caption.getBorderColor());  
 paint.setStyle(Paint.Style.*FILL\_AND\_STROKE*);  
 }  
  
  
 *// set text width to canvas width minus 16dp padding* int textWidth = canvas.getWidth() - (int) (16 \* scale);  
  
 *// init StaticLayout for text* StaticLayout textLayout = new StaticLayout(  
 textString, paint, textWidth, Layout.Alignment.*ALIGN\_CENTER*, 1.0f, 0.0f, false);  
  
 *// get height of multiline text* int textHeight = textLayout.getHeight();  
  
 *// get position of text in the canvas, this will depend in its internal location mode* MemeConfig.Point where = caption.getPositionInCanvas(  
 bitmap.getWidth(), bitmap.getHeight(), textWidth, textHeight);  
  
 *// draw text to the Canvas center* canvas.save();  
 canvas.translate(where.x, where.y);  
 textLayout.draw(canvas);  
  
 *// new antialiased Paint* paint.setColor(caption.getTextColor());  
 paint.setStyle(Paint.Style.*FILL*);  
  
 *// init StaticLayout for text* textLayout = new StaticLayout(  
 textString, paint, textWidth, Layout.Alignment.*ALIGN\_CENTER*, 1.0f, 0.0f, false);  
  
 *// get height of multiline text* textHeight = textLayout.getHeight();  
  
 *// draw text to the Canvas center* textLayout.draw(canvas);  
 canvas.restore();  
 }  
  
 return bitmap;  
 }  
  
 @OnTextChanged(value = R.id.*create\_caption*, callback = OnTextChanged.Callback.*TEXT\_CHANGED*)  
 public void onCaptionChanged(CharSequence text) {  
 if (\_isBottom) {  
 \_memeEditorElements.getCaptionBottom().setText(text.toString());  
 } else {  
 \_memeEditorElements.getCaptionTop().setText(text.toString());  
 }  
 onMemeEditorObjectChanged();  
 }  
  
 public void onMemeEditorObjectChanged() {  
 \_imageEditView.setImageBitmap(null);  
 if (\_lastBitmap != null) {  
 \_lastBitmap.recycle();  
 }  
 Bitmap bmp = renderMemeImageFromElements(this, \_memeEditorElements);  
 \_imageEditView.setImageBitmap(bmp);  
 \_lastBitmap = bmp;  
 }  
  
 *// createForSaving == true will make template text elements empty* public void recreateImage(boolean createForSaving) {  
 if (createForSaving) {  
 for (MemeEditorElements.EditorCaption caption : \_memeEditorElements.getCaptions()) {  
 if (TextUtils.*isEmpty*(caption.getText())) {  
 caption.setText(" ");  
 }  
 }  
 }  
 onMemeEditorObjectChanged();  
 }  
  
 *//////////////////////////////////////////////////////////////  
 ////  
 /// Visibility etc  
 //* @OnClick(R.id.*done\_caption*)  
 public void settingsDone() {  
 \_editBar.setVisibility(View.*GONE*);  
 ActivityUtils.*get*(this).hideSoftKeyboard();  
 onMemeEditorObjectChanged();  
 }  
  
 @OnClick(R.id.*memecreate\_\_moar\_controls\_\_layout*)  
 void onBottomContainerClicked() {  
 toggleMoarControls(true, false);  
 }  
  
 @Override  
 public void onBackPressed() {  
 boolean hasTextInput = !\_create\_caption.getText().toString().isEmpty() ||  
 !\_memeEditorElements.getCaptionBottom().getText().isEmpty() ||  
 !\_memeEditorElements.getCaptionTop().getText().isEmpty();  
  
 if (\_fullscreenImageView.getVisibility() == View.*VISIBLE*) {  
 \_fullscreenImageView.setVisibility(View.*INVISIBLE*);  
 toggleMoarControls(true, false);  
 return;  
 }  
  
 *// Close views above* if (\_bottomContainerVisible) {  
 toggleMoarControls(true, false);  
 return;  
 }  
  
 if (\_editBar.getVisibility() != View.*GONE*) {  
 settingsDone();  
 return;  
 }  
  
 *// Auto save if option checked* if (hasTextInput && \_app.settings.isAutoSaveMeme()) {  
 if (saveMemeToFilesystem(false)) {  
 finish();  
 return;  
 }  
 }  
  
 *// Close if no input* if (!hasTextInput) {  
 finish();  
 return;  
 }  
  
 *// Else wait for double back-press* if (*\_doubleBackToExitPressedOnce*) {  
 super.onBackPressed();  
 return;  
 }  
 *\_doubleBackToExitPressedOnce* = true;  
 Snackbar.*make*(findViewById(android.R.id.*content*), R.string.*press\_back\_again\_to\_stop\_editing\_\_appspecific*, Snackbar.*LENGTH\_SHORT*).show();  
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 *\_doubleBackToExitPressedOnce* = false;  
 }  
 }, 2000);  
 }  
  
 public void touchTopElement() {  
 onImageTouched(\_imageEditView, MotionEvent.*obtain*(1, 1, MotionEvent.*ACTION\_DOWN*, 0, 0, 0));  
 }  
  
 @OnTouch(R.id.*memecreate\_\_activity\_\_image*)  
 public boolean onImageTouched(View view, MotionEvent event) {  
 if (\_editBar.getVisibility() == View.*VISIBLE* && !\_create\_caption.getText().toString().isEmpty()) {  
 onMemeEditorObjectChanged();  
 }  
 if (event.getAction() == MotionEvent.*ACTION\_DOWN*) {  
 float heightOfPic = view.getMeasuredHeight();  
 float heightOfEvent = event.getY();  
  
 int position = (int) (heightOfEvent / heightOfPic \* 100);  
  
 \_isBottom = position >= 50;  
  
 \_editBar.setVisibility(View.*VISIBLE*);  
  
 String \_areaCaption = \_isBottom ?  
 \_memeEditorElements.getCaptionBottom().getText() :  
 \_memeEditorElements.getCaptionTop().getText();  
  
 \_create\_caption.setText(\_areaCaption);  
 \_create\_caption.requestFocus();  
  
 ActivityUtils.*get*(this).showSoftKeyboard();  
  
 if (\_bottomContainerVisible) {  
 toggleMoarControls(true, false);  
 }  
 return true;  
 }  
 return super.onTouchEvent(event);  
 }  
}**

**package net.gsantner.memetastic.activity;  
  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.support.v4.content.LocalBroadcastManager;  
import android.support.v7.widget.GridLayoutManager;  
import android.support.v7.widget.LinearLayoutManager;  
import android.support.v7.widget.RecyclerView;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.LinearLayout;  
import android.widget.TextView;  
  
import net.gsantner.memetastic.App;  
import net.gsantner.memetastic.data.MemeData;  
import net.gsantner.memetastic.ui.GridDecoration;  
import net.gsantner.memetastic.ui.MemeItemAdapter;  
import net.gsantner.memetastic.util.AppCast;  
import net.gsantner.memetastic.util.AppSettings;  
import net.gsantner.memetastic.util.ContextUtils;  
  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
  
import butterknife.BindView;  
import butterknife.ButterKnife;  
import butterknife.Unbinder;  
import io.github.gsantner.memetastic.R;  
  
  
public class MemeFragment extends Fragment {  
 @BindView(R.id.*meme\_fragment\_\_recycler\_view*)  
 RecyclerView \_recyclerMemeList;  
  
 @BindView(R.id.*meme\_fragment\_\_list\_empty\_layout*)  
 LinearLayout \_emptylistLayout;  
  
 @BindView(R.id.*meme\_fragment\_\_list\_empty\_text*)  
 TextView \_emptylistText;  
  
 App \_app;  
 int \_tabPos;  
 String[] \_tagKeys, \_tagValues;  
 private Unbinder \_unbinder;  
 private List<MemeData.Image> \_imageList;  
 private MemeItemAdapter \_recyclerMemeAdapter;  
  
  
 public MemeFragment() {  
 *// Required empty public constructor* }  
  
 *// newInstance constructor for creating fragment with arguments* public static MemeFragment newInstance(int pagePos) {  
 MemeFragment fragmentFirst = new MemeFragment();  
 Bundle args = new Bundle();  
 args.putInt("pos", pagePos);  
 fragmentFirst.setArguments(args);  
 return fragmentFirst;  
 }  
  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 \_app = (App) getActivity().getApplication();  
 \_tabPos = getArguments().getInt("pos");  
  
 \_imageList = new ArrayList<>();  
 }  
  
 private void reloadAdapter() {  
 \_tagKeys = getResources().getStringArray(R.array.*meme\_tags\_\_keys*);  
 \_tagValues = getResources().getStringArray(R.array.*meme\_tags\_\_titles*);  
 if (\_tabPos >= 0 && \_tabPos < \_tagKeys.length) {  
 \_imageList = MemeData.*getImagesWithTag*(\_tagKeys[\_tabPos]);  
 }  
  
 if (\_app.settings.isShuffleTagLists()) {  
 Collections.*shuffle*(\_imageList);  
 }  
  
 List<MemeData.Image> hiddenImages = new ArrayList<>();  
 for (MemeData.Image image : \_imageList) {  
 if (\_app.settings.isHidden(image.fullPath.getAbsolutePath())) {  
 hiddenImages.add(image);  
 }  
 }  
 \_imageList.removeAll(hiddenImages);  
 \_recyclerMemeAdapter.setOriginalImageDataList(\_imageList);  
 \_recyclerMemeAdapter.notifyDataSetChanged();  
 setRecyclerMemeListAdapter(\_recyclerMemeAdapter);  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 *// Inflate the layout for this fragment* View root = inflater.inflate(R.layout.*fragment\_meme*, container, false);  
 \_unbinder = ButterKnife.*bind*(this, root);  
  
  
 \_recyclerMemeList.setHasFixedSize(true);  
 \_recyclerMemeList.setItemViewCacheSize(\_app.settings.getGridColumnCountPortrait() \* \_app.settings.getGridColumnCountLandscape() \* 2);  
 \_recyclerMemeList.setDrawingCacheEnabled(true);  
 \_recyclerMemeList.setDrawingCacheQuality(View.*DRAWING\_CACHE\_QUALITY\_LOW*);  
 \_recyclerMemeList.addItemDecoration(new GridDecoration(1.7f));  
  
 int a = AppSettings.*get*().getMemeListViewType();  
 if (AppSettings.*get*().getMemeListViewType() == MemeItemAdapter.*VIEW\_TYPE\_\_ROWS\_WITH\_TITLE*) {  
 RecyclerView.LayoutManager recyclerLinearLayout = new LinearLayoutManager(getActivity(), LinearLayoutManager.*VERTICAL*, false);  
 \_recyclerMemeList.setLayoutManager(recyclerLinearLayout);  
 } else {  
 int gridColumns = ContextUtils.*get*().isInPortraitMode()  
 ? \_app.settings.getGridColumnCountPortrait()  
 : \_app.settings.getGridColumnCountLandscape();  
 RecyclerView.LayoutManager recyclerGridLayout = new GridLayoutManager(getActivity(), gridColumns);  
  
 \_recyclerMemeList.setLayoutManager(recyclerGridLayout);  
 }  
  
 \_emptylistText.setText(getString(R.string.*no\_custom\_templates\_description\_\_appspecific*, getString(R.string.*custom\_templates\_visual*)));  
 \_recyclerMemeAdapter = new MemeItemAdapter(\_imageList, getActivity(), AppSettings.*get*().getMemeListViewType());  
 setRecyclerMemeListAdapter(\_recyclerMemeAdapter);  
  
 return root;  
 }  
  
 private void setRecyclerMemeListAdapter(MemeItemAdapter adapter) {  
 adapter.setFilter("");  
 \_recyclerMemeList.setAdapter(adapter);  
 boolean isEmpty = adapter.getItemCount() == 0;  
 \_emptylistLayout.setVisibility(isEmpty ? View.*VISIBLE* : View.*GONE*);  
 \_recyclerMemeList.setVisibility(isEmpty ? View.*GONE* : View.*VISIBLE*);  
 }  
  
 private BroadcastReceiver \_localBroadcastReceiver = new BroadcastReceiver() {  
 @SuppressWarnings("unchecked")  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 String action = intent.getAction();  
 switch (action) {  
 case AppCast.ASSETS\_LOADED.*ACTION*: {  
 reloadAdapter();  
 return;  
 }  
 }  
 }  
 };  
  
 @Override  
 public void onResume() {  
 super.onResume();  
 LocalBroadcastManager.*getInstance*(getActivity()).registerReceiver(\_localBroadcastReceiver, AppCast.*getLocalBroadcastFilter*());  
 reloadAdapter();  
 }  
  
 @Override  
 public void onPause() {  
 super.onPause();  
 LocalBroadcastManager.*getInstance*(getActivity()).unregisterReceiver(\_localBroadcastReceiver);  
 }  
  
 @Override  
 public void onDestroy() {  
 super.onDestroy();  
 if (\_unbinder != null) {  
 \_unbinder.unbind();  
 }  
 }  
  
  
}**

**package net.gsantner.memetastic.activity;  
  
import android.support.v4.app.Fragment;  
import android.support.v4.app.FragmentManager;  
import android.support.v4.app.FragmentPagerAdapter;  
import android.support.v4.view.PagerAdapter;  
  
public class MemePagerAdapter extends FragmentPagerAdapter {  
 int \_totalCount;  
 String[] \_pageTitles;  
  
  
 public MemePagerAdapter(FragmentManager fm, int totalCount, String[] pageTitles) {  
 super(fm);  
 \_totalCount = totalCount;  
 \_pageTitles = pageTitles;  
 }  
  
 @Override  
 public CharSequence getPageTitle(int position) {  
 return \_pageTitles[position];  
 }  
  
 @Override  
 public Fragment getItem(int i) {  
 return MemeFragment.*newInstance*(i);  
 }  
  
 @Override  
 public int getCount() {  
 return \_totalCount;  
 }  
  
 @Override  
 public int getItemPosition(Object object) {  
 return PagerAdapter.*POSITION\_NONE*;  
 }  
}**

**package net.gsantner.memetastic.activity;  
  
import android.content.Context;  
import android.content.Intent;  
import android.net.Uri;  
import android.support.v7.preference.Preference;  
import android.support.v7.preference.PreferenceGroup;  
  
import net.gsantner.memetastic.util.AppSettings;  
import net.gsantner.opoc.format.markdown.SimpleMarkdownParser;  
import net.gsantner.opoc.preference.GsPreferenceFragmentCompat;  
import net.gsantner.opoc.util.ActivityUtils;  
import net.gsantner.opoc.util.ShareUtil;  
  
import java.io.IOException;  
import java.util.Locale;  
  
import io.github.gsantner.memetastic.R;  
  
public class MoreInfoFragment extends GsPreferenceFragmentCompat<AppSettings> {  
 public static final String *TAG* = "MoreInfoFragment";  
  
 public static MoreInfoFragment newInstance() {  
 return new MoreInfoFragment();  
 }  
  
 @Override  
 public int getPreferenceResourceForInflation() {  
 return R.xml.*prefactions\_\_more\_information*;  
 }  
  
 @Override  
 public String getFragmentTag() {  
 return *TAG*;  
 }  
  
 @Override  
 protected AppSettings getAppSettings(Context context) {  
 return \_appSettings != null ? \_appSettings : new AppSettings(context);  
 }  
  
 @Override  
 @SuppressWarnings({"ConstantConditions", "ConstantIfStatement", "StatementWithEmptyBody"})  
 public Boolean onPreferenceClicked(Preference preference, String key, int keyResId) {  
 ActivityUtils au = new ActivityUtils(getActivity());  
 if (isAdded() && preference.hasKey()) {  
 switch (keyToStringResId(preference)) {  
 case R.string.*pref\_key\_\_more\_info\_\_app*: {  
 \_cu.openWebpageInExternalBrowser(getString(R.string.*app\_web\_url*));  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_settings*: {  
 au.animateToActivity(SettingsActivity.class, false, 124);  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_rate\_app*: {  
 au.showGooglePlayEntryForThisApp();  
 return true;  
 }  
  
 case R.string.*pref\_key\_\_more\_info\_\_join\_community*: {  
 \_cu.openWebpageInExternalBrowser(getString(R.string.*app\_community\_url*));  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_help*: {  
 \_cu.openWebpageInExternalBrowser(  
 String.*format*("https://gsantner.net/project/%s.html?source=inapp\_more\_help\_faq",  
 getString(R.string.*app\_name\_real*).toLowerCase()));  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_donate*: {  
 \_cu.openWebpageInExternalBrowser(getString(R.string.*app\_donate\_url*));  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_bug\_reports*: {  
 \_cu.openWebpageInExternalBrowser(getString(R.string.*app\_bugreport\_url*));  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_translate*: {  
 \_cu.openWebpageInExternalBrowser(getString(R.string.*app\_translate\_url*));  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_project\_contribution\_info*: {  
 \_cu.openWebpageInExternalBrowser(getString(R.string.*app\_contribution\_info\_url*));  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_source\_code*: {  
 \_cu.openWebpageInExternalBrowser(getString(R.string.*app\_source\_code\_url*));  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_project\_license*: {  
 au.showDialogWithRawFileInWebView("license.txt", R.string.*project\_license*);  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_open\_source\_licenses*: {  
 try {  
 au.showDialogWithHtmlTextView(R.string.*licenses*, new SimpleMarkdownParser().parse(  
 getResources().openRawResource(R.raw.*licenses\_3rd\_party*),  
 "", SimpleMarkdownParser.*FILTER\_ANDROID\_TEXTVIEW*).getHtml());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_contributors\_public\_info*: {  
 try {  
 au.showDialogWithHtmlTextView(R.string.*contributors*, new SimpleMarkdownParser().parse(  
 getResources().openRawResource(R.raw.*contributors*),  
 "", SimpleMarkdownParser.*FILTER\_ANDROID\_TEXTVIEW*).getHtml());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return true;  
 }  
 case R.string.*pref\_key\_\_more\_info\_\_copy\_build\_information*: {  
 new ShareUtil(getContext()).setClipboard(preference.getSummary());  
 SimpleMarkdownParser smp = new SimpleMarkdownParser();  
 try {  
 String html = smp.parse(getResources().openRawResource(R.raw.*changelog*), "", SimpleMarkdownParser.*FILTER\_ANDROID\_TEXTVIEW*, SimpleMarkdownParser.*FILTER\_CHANGELOG*).getHtml();  
 au.showDialogWithHtmlTextView(R.string.*changelog*, html);  
 } catch (Exception ex) {  
  
 }  
 return true;  
 }  
 }  
 }  
 return null;  
 }  
  
 @Override  
 protected boolean isAllowedToTint(Preference pref) {  
 return !getString(R.string.*pref\_key\_\_more\_info\_\_app*).equals(pref.getKey());  
 }  
  
 @Override  
 public synchronized void doUpdatePreferences() {  
 super.doUpdatePreferences();  
 Context context = getContext();  
 if (context == null) {  
 return;  
 }  
 Locale locale = Locale.*getDefault*();  
 String tmp;  
 Preference pref;  
 updateSummary(R.string.*pref\_key\_\_more\_info\_\_project\_license*, getString(R.string.*app\_license\_name*));  
  
 *// Basic app info* if ((pref = findPreference(R.string.*pref\_key\_\_more\_info\_\_app*)) != null && pref.getSummary() == null) {  
 pref.setIcon(R.drawable.*ic\_launcher*);  
 pref.setSummary(String.*format*(locale, "%s\nVersion v%s (%d)", \_cu.getPackageIdReal(), \_cu.getAppVersionName(), \_cu.bcint("VERSION\_CODE", 0)));  
 }  
  
 *// Extract some build information and publish in summary* if ((pref = findPreference(R.string.*pref\_key\_\_more\_info\_\_copy\_build\_information*)) != null && pref.getSummary() == null) {  
 String summary = String.*format*(locale, "\n<b>Package:</b> %s\n<b>Version:</b> v%s (%d)", \_cu.getPackageIdReal(), \_cu.getAppVersionName(), \_cu.bcint("VERSION\_CODE", 0));  
 summary += (tmp = \_cu.bcstr("FLAVOR", "")).isEmpty() ? "" : ("\n<b>Flavor:</b> " + tmp.replace("flavor", ""));  
 summary += (tmp = \_cu.bcstr("BUILD\_TYPE", "")).isEmpty() ? "" : (" (" + tmp + ")");  
 summary += (tmp = \_cu.bcstr("BUILD\_DATE", "")).isEmpty() ? "" : ("\n<b>Build date:</b> " + tmp);  
 summary += (tmp = \_cu.getAppInstallationSource()).isEmpty() ? "" : ("\n<b>ISource:</b> " + tmp);  
 summary += (tmp = \_cu.bcstr("GITHASH", "")).isEmpty() ? "" : ("\n<b>VCS Hash:</b> " + tmp);  
 pref.setSummary(\_cu.htmlToSpanned(summary.trim().replace("\n", "<br/>")));  
 }  
  
 *// Extract project team from raw ressource, where 1 person = 4 lines  
 // 1) Name/Title, 2) Description/Summary, 3) Link/View-Intent, 4) Empty line* if ((pref = findPreference(R.string.*pref\_key\_\_more\_info\_\_project\_team*)) != null && ((PreferenceGroup) pref).getPreferenceCount() == 0) {  
 String[] data = (\_cu.readTextfileFromRawRes(R.raw.*project\_team*, "", "").trim() + "\n\n").split("\n");  
 for (int i = 0; i + 2 < data.length; i += 4) {  
 Preference person = new Preference(context);  
 person.setTitle(data[i]);  
 person.setSummary(data[i + 1]);  
 person.setIcon(R.drawable.*ic\_person\_black\_24dp*);  
 try {  
 Uri uri = Uri.*parse*(data[i + 2]);  
 Intent intent = new Intent(Intent.*ACTION\_VIEW*, uri);  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 person.setIntent(intent);  
 } catch (Exception ignored) {  
 }  
 appendPreference(person, (PreferenceGroup) pref);  
 }  
 }  
 if ((pref = findPreference(R.string.*pref\_key\_\_more\_info\_\_help*)) != null) {  
 pref.setTitle(getString(R.string.*help*) + " / FAQ");  
 }  
 }  
}**

**package net.gsantner.memetastic.activity;  
  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.support.design.widget.AppBarLayout;  
import android.support.v4.app.FragmentTransaction;  
import android.support.v7.app.AppCompatActivity;  
import android.support.v7.preference.Preference;  
import android.support.v7.preference.PreferenceFragmentCompat;  
import android.support.v7.preference.PreferenceScreen;  
import android.support.v7.widget.Toolbar;  
import android.text.TextUtils;  
import android.view.View;  
  
import net.gsantner.memetastic.service.AssetUpdater;  
import net.gsantner.memetastic.service.ThumbnailCleanupTask;  
import net.gsantner.memetastic.util.AppSettings;  
import net.gsantner.memetastic.util.MediaStoreUtils;  
import net.gsantner.memetastic.util.PermissionChecker;  
import net.gsantner.opoc.preference.GsPreferenceFragmentCompat;  
import net.gsantner.opoc.preference.SharedPreferencesPropertyBackend;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.Date;  
  
import butterknife.BindView;  
import butterknife.ButterKnife;  
import io.github.gsantner.memetastic.R;  
  
public class SettingsActivity extends AppCompatActivity {  
 static final int *ACTIVITY\_ID* = 10;  
  
 static class RESULT {  
 static final int *NOCHANGE* = -1;  
 static final int *CHANGE* = 1;  
 static final int *CHANGE\_RESTART* = 2;  
 }  
  
 @BindView(R.id.*settings\_\_appbar*)  
 protected AppBarLayout appBarLayout;  
 @BindView(R.id.*settings\_\_toolbar*)  
 protected Toolbar toolbar;  
  
 private AppSettings appSettings;  
 public static int *activityRetVal* = RESULT.*NOCHANGE*;  
  
 public void onCreate(Bundle b) {  
 super.onCreate(b);  
 setContentView(R.layout.*settings\_\_activity*);  
 ButterKnife.*bind*(this);  
 toolbar.setTitle(R.string.*settings*);  
 setSupportActionBar(toolbar);  
 appSettings = AppSettings.*get*();  
 toolbar.setNavigationIcon(getResources().getDrawable(R.drawable.*ic\_arrow\_back\_white\_48px*));  
 toolbar.setNavigationOnClickListener(new View.OnClickListener() {  
 public void onClick(View v) {  
 SettingsActivity.this.onBackPressed();  
 }  
 });  
 *activityRetVal* = RESULT.*NOCHANGE*;  
 showFragment(SettingsFragmentMaster.*TAG*, false);  
 }  
  
 @Override  
 public void onBackPressed() {  
 GsPreferenceFragmentCompat prefFrag = (GsPreferenceFragmentCompat) getSupportFragmentManager().findFragmentByTag(SettingsFragmentMaster.*TAG*);  
 if (prefFrag != null && prefFrag.canGoBack()) {  
 prefFrag.goBack();  
 return;  
 }  
 super.onBackPressed();  
 }  
  
 protected void showFragment(String tag, boolean addToBackStack) {  
 GsPreferenceFragmentCompat fragment = (GsPreferenceFragmentCompat) getSupportFragmentManager().findFragmentByTag(tag);  
 if (fragment == null) {  
 switch (tag) {  
 case SettingsFragmentMaster.*TAG*:  
 default:  
 fragment = new SettingsFragmentMaster();  
 toolbar.setTitle(R.string.*settings*);  
 break;  
 }  
 }  
 FragmentTransaction t = getSupportFragmentManager().beginTransaction();  
 if (addToBackStack) {  
 t.addToBackStack(tag);  
 }  
 t.replace(R.id.*settings\_\_fragment\_container*, fragment, tag).commit();  
 }  
  
 @Override  
 protected void onStop() {  
 setResult(*activityRetVal*);  
 super.onStop();  
 }  
  
 public static class SettingsFragmentMaster extends GsPreferenceFragmentCompat {  
 public static final String *TAG* = "SettingsFragmentMaster";  
  
 @Override  
 protected void onPreferenceChanged(SharedPreferences prefs, String key) {  
 if (*activityRetVal* == RESULT.*NOCHANGE*) {  
 *activityRetVal* = RESULT.*CHANGE*;  
 }  
 }  
  
 @Override  
 public int getPreferenceResourceForInflation() {  
 return R.xml.*preferences\_master*;  
 }  
  
 @Override  
 public String getFragmentTag() {  
 return *TAG*;  
 }  
  
 @Override  
 protected SharedPreferencesPropertyBackend getAppSettings(Context context) {  
 return new AppSettings(context);  
 }  
  
 @SuppressLint("ApplySharedPref")  
 @Override  
 public Boolean onPreferenceClicked(Preference preference, String key, int keyResId) {  
 if (isAdded() && preference.hasKey()) {  
 Context context = getActivity();  
 AppSettings settings = AppSettings.*get*();  
  
  
 if (eq(key, R.string.*pref\_key\_\_memelist\_view\_type*)) {  
 *activityRetVal* = RESULT.*CHANGE\_RESTART*;  
 }  
 if (eq(key, R.string.*pref\_key\_\_cleanup\_thumbnails*)) {  
 new ThumbnailCleanupTask(context).start();  
 return true;  
 }  
 if (eq(key, R.string.*pref\_key\_\_is\_overview\_statusbar\_hidden*)) {  
 *activityRetVal* = RESULT.*CHANGE\_RESTART*;  
 }  
 if (eq(key, R.string.*pref\_key\_\_language*)) {  
 *activityRetVal* = RESULT.*CHANGE\_RESTART*;  
 }  
 if (eq(key, R.string.*pref\_key\_\_download\_assets\_try*)) {  
 if (PermissionChecker.*doIfPermissionGranted*(getActivity())) {  
 Date zero = new Date(0);  
 settings.setLastArchiveCheckDate(zero);  
 settings.setLastArchiveDate(zero);  
 settings.getDefaultPreferences().edit().commit();  
 new AssetUpdater.UpdateThread(context, true).start();  
 getActivity().finish();  
 }  
 }  
 if (eq(key, R.string.*pref\_key\_\_is\_show\_in\_gallery*)) {  
 boolean showInGallery = settings.getDefaultPreferences().getBoolean(key, true);  
 File memeDirectory = AssetUpdater.*getMemesDir*(AppSettings.*get*());  
 File noMediaFile = new File(memeDirectory, ".nomedia");  
 if (showInGallery) {  
 noMediaFile.delete();  
 MediaStoreUtils.*deleteFileFromMediaStore*(context, noMediaFile);  
 File[] files = memeDirectory.listFiles();  
 for (int i = 0; i < files.length; i++) {  
 MediaStoreUtils.*deleteFileFromMediaStore*(context, files[i]);  
 MediaStoreUtils.*addFileToMediaStore*(context, files[i]);  
 }  
 } else {  
 try {  
 noMediaFile.createNewFile();  
 MediaStoreUtils.*addFileToMediaStore*(context, noMediaFile);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 return null;  
 }  
  
 @Override  
 protected void onPreferenceScreenChanged(PreferenceFragmentCompat preferenceFragmentCompat, PreferenceScreen preferenceScreen) {  
 super.onPreferenceScreenChanged(preferenceFragmentCompat, preferenceScreen);  
 if (!TextUtils.*isEmpty*(preferenceScreen.getTitle())) {  
 SettingsActivity a = (SettingsActivity) getActivity();  
 if (a != null) {  
 a.toolbar.setTitle(preferenceScreen.getTitle());  
 }  
 }  
 }  
  
 @Override  
 public synchronized void doUpdatePreferences() {  
 super.doUpdatePreferences();  
 setPreferenceVisible(R.string.*pref\_key\_\_download\_assets\_try*, false);  
 }  
 }  
}**

**public class MainActivity extends AppCompatActivity  
 implements NavigationView.OnNavigationItemSelectedListener, ViewPager.OnPageChangeListener, BottomNavigationView.OnNavigationItemSelectedListener {  
 public static final int *REQUEST\_LOAD\_GALLERY\_IMAGE* = 50;  
 public static final int *REQUEST\_TAKE\_CAMERA\_PICTURE* = 51;  
 public static final int *REQUEST\_SHOW\_IMAGE* = 52;  
 public static final String *IMAGE\_PATH* = "imagePath";  
 public static final String *IMAGE\_POS* = "image\_pos";  
 public static final boolean *LOCAL\_ONLY\_MODE* = true;  
 public static final boolean *DISABLE\_ONLINE\_ASSETS* = true;  
  
 private static boolean *\_isShowingFullscreenImage* = false;  
  
 @BindView(R.id.*toolbar*)  
 Toolbar \_toolbar;  
  
 @BindView(R.id.*bottom\_navigation\_bar*)  
 BottomNavigationView \_bottomNav;  
 private MenuItem \_lastBottomMenuItem;  
  
 @BindView(R.id.*main\_\_tabs*)  
 TabLayout \_tabLayout;  
  
 @BindView(R.id.*main\_\_more\_info\_fragment\_container*)  
 LinearLayout \_moreInfoContainer;  
  
 @BindView(R.id.*main\_activity\_\_placeholder*)  
 FrameLayout \_placeholder;  
  
 @BindView(R.id.*main\_activity\_\_view\_pager*)  
 ViewPager \_viewPager;  
  
 @BindView(R.id.*main\_\_activity\_\_recycler\_view*)  
 RecyclerView \_recyclerMemeList;  
  
 @BindView(R.id.*main\_\_activity\_\_list\_empty\_\_layout*)  
 LinearSplitLayout \_emptylistLayout;  
  
 @BindView(R.id.*main\_\_activity\_\_list\_empty\_\_text*)  
 TextView \_emptylistText;  
  
 @BindView(R.id.*main\_\_activity\_\_infobar*)  
 LinearLayout \_infoBar;  
  
 @BindView(R.id.*main\_\_activity\_\_infobar\_\_progress*)  
 ProgressBar \_infoBarProgressBar;  
  
 @BindView(R.id.*main\_\_activity\_\_infobar\_\_image*)  
 ImageView \_infoBarImage;  
  
 @BindView(R.id.*main\_\_activity\_\_infobar\_\_text*)  
 TextView \_infoBarText;  
  
 App app;  
 private AppSettings \_appSettings;  
 private ActivityUtils \_activityUtils;  
 private String cameraPictureFilepath = "";  
 String[] \_tagKeys, \_tagValues;  
 private int \_currentMainMode = 0;  
 private long \_lastInfoBarTextShownAt = 0;  
 private SearchView \_searchView;  
 private MenuItem \_searchItem;  
 private String \_currentSearch = "";  
  
 private static final String *BOTTOM\_NAV\_POSITION* = "bottom\_nav\_position";  
  
 @SuppressLint("ClickableViewAccessibility")  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 if (savedInstanceState != null) {  
 \_currentMainMode = savedInstanceState.getInt(*BOTTOM\_NAV\_POSITION*);  
 }  
 \_appSettings = new AppSettings(this);  
 \_activityUtils = new ActivityUtils(this);  
 \_activityUtils.setAppLanguage(\_appSettings.getLanguage());  
 if (\_appSettings.isOverviewStatusBarHidden()) {  
 getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*, WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);  
 }  
 setContentView(R.layout.*main\_\_activity*);  
  
 *// Bind UI* app = (App) getApplication();  
 ButterKnife.*bind*(this);  
  
 *// Setup \_toolbar* setSupportActionBar(\_toolbar);  
  
 \_tagKeys = getResources().getStringArray(R.array.*meme\_tags\_\_keys*);  
 \_tagValues = getResources().getStringArray(R.array.*meme\_tags\_\_titles*);  
  
 if (MainActivity.*LOCAL\_ONLY\_MODE*) {  
 for (int i = 0; i < \_tagKeys.length; i++) {  
 \_tagKeys[i] = "other";  
 }  
 \_tagKeys = new String[]{\_tagKeys[0]};  
 \_tagValues = new String[]{\_tagValues[0]};  
 }  
  
  
 \_recyclerMemeList.setHasFixedSize(true);  
 \_recyclerMemeList.setItemViewCacheSize(\_appSettings.getGridColumnCountPortrait() \* \_appSettings.getGridColumnCountLandscape() \* 2);  
 \_recyclerMemeList.setDrawingCacheEnabled(true);  
 \_recyclerMemeList.setDrawingCacheQuality(View.*DRAWING\_CACHE\_QUALITY\_LOW*);  
 \_recyclerMemeList.addItemDecoration(new GridDecoration(1.7f));  
  
 if (\_appSettings.getMemeListViewType() == MemeItemAdapter.*VIEW\_TYPE\_\_ROWS\_WITH\_TITLE*) {  
 RecyclerView.LayoutManager recyclerLinearLayout = new LinearLayoutManager(this, LinearLayoutManager.*VERTICAL*, false);  
 \_recyclerMemeList.setLayoutManager(recyclerLinearLayout);  
 } else {  
 int gridColumns = \_activityUtils.isInPortraitMode()  
 ? \_appSettings.getGridColumnCountPortrait()  
 : \_appSettings.getGridColumnCountLandscape();  
 RecyclerView.LayoutManager recyclerGridLayout = new GridLayoutManager(this, gridColumns);  
  
 \_recyclerMemeList.setLayoutManager(recyclerGridLayout);  
 }  
  
 for (String cat : \_tagValues) {  
 TabLayout.Tab tab = \_tabLayout.newTab();  
 tab.setText(cat);  
 \_tabLayout.addTab(tab);  
 }  
  
 *// Basically enable "other" only mode* if (MainActivity.*LOCAL\_ONLY\_MODE*) {  
 \_tabLayout.setVisibility(View.*GONE*);  
 }  
 *// END* \_viewPager.setOffscreenPageLimit(5);  
 \_viewPager.setAdapter(new MemePagerAdapter(getSupportFragmentManager(), \_tagKeys.length, \_tagValues));  
 \_tabLayout.setupWithViewPager(\_viewPager);  
 selectTab(app.settings.getLastSelectedTab(), app.settings.getDefaultMainMode());  
 \_infoBarProgressBar.getProgressDrawable().setColorFilter(ContextCompat.*getColor*(this, R.color.*accent*), PorterDuff.Mode.*SRC\_IN*);  
  
  
 *// Show first start dialog / changelog* try {  
 if (\_appSettings.isAppCurrentVersionFirstStart(true)) {  
 SimpleMarkdownParser smp = SimpleMarkdownParser.*get*().setDefaultSmpFilter(SimpleMarkdownParser.*FILTER\_ANDROID\_TEXTVIEW*);  
 String html = "";  
 html += smp.parse(getString(R.string.*copyright\_license\_text\_official*).replace("\n", " \n"), "").getHtml();  
 html += "<br/><br/><br/><big><big>" + getString(R.string.*changelog*) + "</big></big><br/>" + smp.parse(getResources().openRawResource(R.raw.*changelog*), "", SimpleMarkdownParser.*FILTER\_ANDROID\_TEXTVIEW*, SimpleMarkdownParser.*FILTER\_CHANGELOG*);  
 html += "<br/><br/><br/><big><big>" + getString(R.string.*licenses*) + "</big></big><br/>" + smp.parse(getResources().openRawResource(R.raw.*licenses\_3rd\_party*), "").getHtml();  
  
 \_activityUtils.showDialogWithHtmlTextView(R.string.*licenses*, html);  
 }  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 new AssetUpdater.LoadAssetsThread(this).start();  
  
 if (PermissionChecker.*doIfPermissionGranted*(this)) {  
 ContextUtils.*checkForAssetUpdates*(this);  
 }  
  
 \_bottomNav.setOnNavigationItemSelectedListener(this);  
 }  
  
 public void updateHiddenNavOption() {  
 MenuItem hiddenItem = \_bottomNav.getMenu().findItem(R.id.*nav\_mode\_hidden*);  
 for (String hidden : app.settings.getHiddenMemesTemplate()) {  
 MemeData.Image image = MemeData.*findImage*(new File(hidden));  
 if (image != null) {  
 hiddenItem.setVisible(true);  
 return;  
 }  
 }  
 hiddenItem.setVisible(false);  
 }  
  
 @SuppressWarnings("ConstantConditions")  
 private void selectTab(int pos, int mainMode) {  
 MenuItem navItem = null;  
 switch (mainMode) {  
 case 0:  
 pos = pos >= 0 ? pos : \_tabLayout.getTabCount() - 1;  
 pos = pos < \_tabLayout.getTabCount() ? pos : 0;  
 \_tabLayout.getTabAt(pos).select();  
 break;  
 case 1:  
 navItem = \_bottomNav.getMenu().findItem(R.id.*nav\_mode\_favs*);  
 break;  
 case 2:  
 navItem = \_bottomNav.getMenu().findItem(R.id.*nav\_mode\_saved*);  
 break;  
 case 3:  
 navItem = \_bottomNav.getMenu().findItem(R.id.*nav\_mode\_hidden*);  
 break;  
 case 4:  
 navItem = \_bottomNav.getMenu().findItem(R.id.*nav\_more*);  
 break;  
 }  
  
 if (navItem != null) {  
 navItem.setChecked(true);  
 onNavigationItemSelected(navItem);  
 }  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 *//new AndroidSupportMeWrapper(this).mainOnResume();* if (MainActivity.*LOCAL\_ONLY\_MODE*) {  
 \_tabLayout.setVisibility(View.*GONE*);  
 }  
 if (*\_isShowingFullscreenImage*) {  
 *\_isShowingFullscreenImage* = false;  
 overridePendingTransition(R.anim.*fadein*, R.anim.*fadeout*);  
 }  
 LocalBroadcastManager.*getInstance*(this).registerReceiver(\_localBroadcastReceiver, AppCast.*getLocalBroadcastFilter*());  
  
 if (SettingsActivity.*activityRetVal* == SettingsActivity.RESULT.*CHANGE\_RESTART*) {  
 SettingsActivity.*activityRetVal* = SettingsActivity.RESULT.*NOCHANGE*;  
 recreate();  
 }  
  
 try {  
 if (new Random().nextInt(10) > 2) {  
 Method m = getClass().getMethod(new String(Base64.*decode*("Z2V0UGFja2FnZU5hbWU=", Base64.*DEFAULT*)));  
 String ret = (String) m.invoke(this);  
 if (!ret.startsWith(new String(Base64.*decode*("bmV0LmdzYW50bmVyLg==", Base64.*DEFAULT*))) && !ret.startsWith(new String(Base64.*decode*("aW8uZ2l0aHViLmdzYW50bmVyLg==", Base64.*DEFAULT*)))) {  
 m = System.class.getMethod(new String(Base64.*decode*("ZXhpdA==", Base64.*DEFAULT*)), int.class);  
 m.invoke(null, 0);  
 }  
 }  
 } catch (Exception ignored) {  
 }  
 \_viewPager.addOnPageChangeListener(this);  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 LocalBroadcastManager.*getInstance*(this).unregisterReceiver(\_localBroadcastReceiver);  
 \_viewPager.removeOnPageChangeListener(this);  
 }  
  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (PermissionChecker.*checkPermissionResult*(this, requestCode, permissions, grantResults)) {  
 ContextUtils.*checkForAssetUpdates*(this);  
 }  
 new AssetUpdater.LoadAssetsThread(this).start();  
 selectTab(\_tabLayout.getSelectedTabPosition(), \_currentMainMode);  
 }  
  
 @Override  
 public void onBackPressed() {  
 if (!\_searchView.isIconified()) {  
 \_searchView.setIconified(true);  
 updateSearchFilter("");  
 } else {  
 super.onBackPressed();  
 }  
 }  
  
 @SuppressWarnings("ResultOfMethodCallIgnored")  
 public boolean handleBarClick(MenuItem item) {  
 List<MemeData.Image> imageList = null;  
  
 switch (item.getItemId()) {  
 case R.id.*action\_picture\_from\_gallery*: {  
 if (PermissionChecker.*doIfPermissionGranted*(this)) {  
 Intent i = new Intent(Intent.*ACTION\_PICK*, MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*);  
 ActivityUtils.*get*(this).animateToActivity(i, false, *REQUEST\_LOAD\_GALLERY\_IMAGE*);  
 }  
 return true;  
 }  
 case R.id.*action\_picture\_from\_camera*: {  
 showCameraDialog();  
 return true;  
 }  
  
 case R.id.*nav\_mode\_create*: {  
 \_currentMainMode = 0;  
 selectTab(app.settings.getLastSelectedTab(), app.settings.getDefaultMainMode());  
 \_toolbar.setTitle(R.string.*app\_name*);  
 break;  
 }  
 case R.id.*nav\_mode\_favs*: {  
 \_currentMainMode = 1;  
 imageList = new ArrayList<>();  
 \_emptylistText.setText(R.string.*no\_favourites\_description\_\_appspecific*);  
 for (String fav : app.settings.getFavoriteMemeTemplates()) {  
 MemeData.Image img = MemeData.*findImage*(new File(fav));  
 if (img != null) {  
 imageList.add(img);  
 }  
 }  
 \_toolbar.setTitle(R.string.*favs*);  
 break;  
 }  
 case R.id.*nav\_mode\_saved*: {  
 \_currentMainMode = 2;  
 \_emptylistText.setText(R.string.*no\_memes\_saved\_description\_\_appspecific*);  
 if (PermissionChecker.*hasExtStoragePerm*(this)) {  
 File folder = AssetUpdater.*getMemesDir*(AppSettings.*get*());  
 folder.mkdirs();  
 imageList = MemeData.*getCreatedMemes*();  
 }  
 \_toolbar.setTitle(R.string.*saved*);  
 break;  
 }  
  
 case R.id.*nav\_mode\_hidden*: {  
 \_currentMainMode = 3;  
 imageList = new ArrayList<>();  
  
 for (String hidden : app.settings.getHiddenMemesTemplate()) {  
 MemeData.Image image = MemeData.*findImage*(new File(hidden));  
 if (image != null) {  
 imageList.add(image);  
 }  
 }  
 \_toolbar.setTitle(R.string.*hidden*);  
 break;  
 }  
 case R.id.*nav\_more*: {  
 \_currentMainMode = 4;  
 \_toolbar.setTitle(R.string.*more*);  
 break;  
 }  
 }  
  
 *// Change mode  
 //\_tabLayout.setVisibility(item.getItemId() == R.id.nav\_mode\_create ? View.VISIBLE : View.GONE);* \_moreInfoContainer.setVisibility(View.*GONE*);  
 if (item.getItemId() == R.id.*nav\_more*) {  
 \_placeholder.setVisibility(View.*GONE*);  
 \_viewPager.setVisibility(View.*GONE*);  
 \_moreInfoContainer.setVisibility(View.*VISIBLE*);  
 } else if (item.getItemId() != R.id.*nav\_mode\_create*) {  
 \_viewPager.setVisibility(View.*GONE*);  
 \_placeholder.setVisibility(View.*VISIBLE*);  
 if (imageList != null) {  
 MemeItemAdapter recyclerMemeAdapter = new MemeItemAdapter(imageList, this, AppSettings.*get*().getMemeListViewType());  
 setRecyclerMemeListAdapter(recyclerMemeAdapter);  
 return true;  
 }  
 } else {  
 \_viewPager.setVisibility(View.*VISIBLE*);  
 \_placeholder.setVisibility(View.*GONE*);  
 }  
  
 return true;  
 }  
  
 private void setRecyclerMemeListAdapter(MemeItemAdapter adapter) {  
 adapter.setFilter(\_currentSearch);  
 \_recyclerMemeList.setAdapter(adapter);  
 boolean isEmpty = adapter.getItemCount() == 0;  
 \_emptylistLayout.setVisibility(isEmpty ? View.*VISIBLE* : View.*GONE*);  
 \_recyclerMemeList.setVisibility(isEmpty ? View.*GONE* : View.*VISIBLE*);  
 }  
  
  
 private void updateSearchFilter(String newFilter) {  
 if (\_currentMainMode != 0) {  
 \_currentSearch = newFilter;  
 if (\_recyclerMemeList.getAdapter() != null) {  
 ((MemeItemAdapter) \_recyclerMemeList.getAdapter()).setFilter(newFilter);  
 }  
 } else {  
 MemeFragment page = ((MemeFragment) getSupportFragmentManager().findFragmentByTag("android:switcher:" + R.id.*main\_activity\_\_view\_pager* + ":" + \_viewPager.getCurrentItem()));  
 if (page != null && page.\_recyclerMemeList.getAdapter() != null) {  
 ((MemeItemAdapter) page.\_recyclerMemeList.getAdapter()).setFilter(newFilter);  
 }  
 }  
 }  
  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
  
 if (requestCode == *REQUEST\_LOAD\_GALLERY\_IMAGE*) {  
 if (resultCode == *RESULT\_OK* && data != null) {  
 Uri selectedImage = data.getData();  
 String[] filePathColumn = {MediaStore.Images.Media.*DATA*};  
 String picturePath = null;  
  
 Cursor cursor = getContentResolver().query(selectedImage, filePathColumn, null, null, null);  
 if (cursor != null && cursor.moveToFirst()) {  
 for (String column : filePathColumn) {  
 int curColIndex = cursor.getColumnIndex(column);  
 if (curColIndex == -1) {  
 continue;  
 }  
 picturePath = cursor.getString(curColIndex);  
 if (!TextUtils.*isEmpty*(picturePath)) {  
 break;  
 }  
 }  
 cursor.close();  
 }  
  
 *// Retrieve image from file descriptor / Cloud, e.g.: Google Drive, Picasa* if (picturePath == null && Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*KITKAT*) {  
 try {  
 ParcelFileDescriptor parcelFileDescriptor = getContentResolver().openFileDescriptor(selectedImage, "r");  
 if (parcelFileDescriptor != null) {  
 FileDescriptor fileDescriptor = parcelFileDescriptor.getFileDescriptor();  
 FileInputStream input = new FileInputStream(fileDescriptor);  
  
 *// Create temporary file in cache directory* picturePath = File.*createTempFile*("image", "tmp", getCacheDir()).getAbsolutePath();  
 FileUtils.*writeFile*(  
 new File(picturePath),  
 FileUtils.*readCloseBinaryStream*(input)  
 );  
 }  
 } catch (IOException e) {  
 *// nothing we can do here, null value will be handled below* }  
 }  
  
 *// Finally check if we got something* if (picturePath == null) {  
 ActivityUtils.*get*(this).showSnackBar(R.string.*error\_couldnot\_load\_picture\_from\_storage*, false);  
 } else {  
 onImageTemplateWasChosen(picturePath);  
 }  
 }  
 }  
  
 if (requestCode == *REQUEST\_TAKE\_CAMERA\_PICTURE*) {  
 if (resultCode == *RESULT\_OK*) {  
 onImageTemplateWasChosen(cameraPictureFilepath);  
 } else {  
 ActivityUtils.*get*(this).showSnackBar(R.string.*error\_picture\_selection*, false);  
 }  
 }  
 if (requestCode == *REQUEST\_SHOW\_IMAGE*) {  
 selectTab(\_tabLayout.getSelectedTabPosition(), \_currentMainMode);  
 }  
 }  
  
 */\*\*  
 \* Show the camera picker via intent  
 \* Source: http://developer.android.com/training/camera/photobasics.html  
 \*/* public void showCameraDialog() {  
 if (!PermissionChecker.*doIfPermissionGranted*(this)) {  
 return;  
 }  
 Intent takePictureIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  
 File photoFile = null;  
 try {  
 *// Create an image file name* String imageFileName = getString(R.string.*app\_name*) + "\_" + System.*currentTimeMillis*();  
 File storageDir = new File(Environment.*getExternalStoragePublicDirectory*(  
 Environment.*DIRECTORY\_DCIM*), "Camera");  
 photoFile = File.*createTempFile*(imageFileName, ".jpg", storageDir);  
  
 *// Save a file: path for use with ACTION\_VIEW intents* cameraPictureFilepath = photoFile.getAbsolutePath();  
  
 } catch (IOException ex) {  
 ActivityUtils.*get*(this).showSnackBar(R.string.*error\_cannot\_start\_camera*, false);  
 }  
  
 *// Continue only if the File was successfully created* if (photoFile != null) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*N*) {  
 Uri uri = FileProvider.*getUriForFile*(this, getString(R.string.*app\_fileprovider*), photoFile);  
 takePictureIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*, uri);  
 } else {  
 takePictureIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*, Uri.*fromFile*(photoFile));  
 }  
 ActivityUtils.*get*(this).animateToActivity(takePictureIntent, false, *REQUEST\_TAKE\_CAMERA\_PICTURE*);  
 }  
 }  
 }  
  
 public void onImageTemplateWasChosen(String filePath) {  
 final Intent intent = new Intent(this, MemeCreateActivity.class);  
 intent.putExtra(MemeCreateActivity.*EXTRA\_IMAGE\_PATH*, filePath);  
 ActivityUtils.*get*(this).animateToActivity(intent, false, MemeCreateActivity.*RESULT\_MEME\_EDITING\_FINISHED*);  
 }  
  
 public void openImageViewActivityWithImage(int pos, String imagePath) {  
 *\_isShowingFullscreenImage* = true;  
  
 Intent intent = new Intent(this, ImageViewActivity.class);  
 intent.putExtra(*IMAGE\_PATH*, imagePath);  
 intent.putExtra(*IMAGE\_POS*, pos);  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_NO\_ANIMATION*);  
 ActivityUtils.*get*(this).animateToActivity(intent, false, *REQUEST\_SHOW\_IMAGE*);  
 }  
  
  
 private BroadcastReceiver \_localBroadcastReceiver = new BroadcastReceiver() {  
 @SuppressWarnings("unchecked")  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 String action = intent.getAction();  
 switch (action) {  
 case AppCast.ASSET\_DOWNLOAD\_REQUEST.*ACTION*: {  
  
 switch (intent.getIntExtra(AppCast.ASSET\_DOWNLOAD\_REQUEST.*EXTRA\_RESULT*, AssetUpdater.UpdateThread.*ASSET\_DOWNLOAD\_REQUEST\_\_FAILED*)) {  
 case AssetUpdater.UpdateThread.*ASSET\_DOWNLOAD\_REQUEST\_\_CHECKING*: {  
 updateInfoBar(0, R.string.*download\_latest\_assets\_checking\_description*, R.drawable.*ic\_file\_download\_white\_32dp*, false);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*ASSET\_DOWNLOAD\_REQUEST\_\_FAILED*: {  
 updateInfoBar(0, R.string.*downloading\_failed*, R.drawable.*ic\_file\_download\_white\_32dp*, false);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*ASSET\_DOWNLOAD\_REQUEST\_\_DO\_DOWNLOAD\_ASK*: {  
 updateInfoBar(0, R.string.*download\_latest\_assets\_checking\_description*, R.drawable.*ic\_file\_download\_white\_32dp*, false);  
 showDownloadDialog();  
 break;  
 }  
 }  
 return;  
 }  
 case AppCast.DOWNLOAD\_STATUS.*ACTION*: {  
 int percent = intent.getIntExtra(AppCast.DOWNLOAD\_STATUS.*EXTRA\_PERCENT*, 100);  
 switch (intent.getIntExtra(AppCast.DOWNLOAD\_STATUS.*EXTRA\_STATUS*, AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_FAILED*)) {  
 case AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_DOWNLOADING*: {  
 updateInfoBar(percent, R.string.*downloading*, R.drawable.*ic\_file\_download\_white\_32dp*, true);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_FAILED*: {  
 updateInfoBar(percent, R.string.*downloading\_failed*, R.drawable.*ic\_mood\_bad\_black\_256dp*, false);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_UNZIPPING*: {  
 updateInfoBar(percent, R.string.*unzipping*, R.drawable.*ic\_file\_download\_white\_32dp*, true);  
 break;  
 }  
 case AssetUpdater.UpdateThread.*DOWNLOAD\_STATUS\_\_FINISHED*: {  
 updateInfoBar(percent, R.string.*successfully\_downloaded*, R.drawable.*ic\_gavel\_white\_48px*, false);  
 break;  
 }  
 }  
 return;  
 }  
 case AppCast.ASSETS\_LOADED.*ACTION*: {  
 selectTab(\_tabLayout.getSelectedTabPosition(), \_currentMainMode);  
 updateHiddenNavOption();  
 break;  
 }  
 }  
 }  
 };  
  
 private void showDownloadDialog() {  
 AlertDialog.Builder dialog = new AlertDialog.Builder(this)  
 .setTitle(R.string.*download\_latest\_assets*)  
 .setMessage(R.string.*download\_latest\_assets\_message\_\_appspecific*)  
 .setNegativeButton(android.R.string.*cancel*, null)  
 .setPositiveButton(android.R.string.*yes*, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialogInterface, int i) {  
 new AssetUpdater.UpdateThread(MainActivity.this, true).start();  
 }  
 });  
 dialog.show();  
 }  
  
 public void updateInfoBar(Integer percent, @StringRes Integer textResId, @DrawableRes Integer image, final boolean showlong) {  
 \_lastInfoBarTextShownAt = System.*currentTimeMillis*();  
 \_infoBar.setVisibility(View.*VISIBLE*);  
 Handler handler = new Handler();  
 handler.postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 if ((System.*currentTimeMillis*() - \_lastInfoBarTextShownAt) > (showlong ? 20 : 2) \* 1000) {  
 \_infoBar.setVisibility(View.*GONE*);  
 }  
 }  
 }, (showlong ? 20 : 2) \* 1000 + 100);  
 if (percent != null) {  
 \_infoBarProgressBar.setProgress(percent);  
 }  
 if (textResId != null) {  
 \_infoBarText.setText(textResId);  
 }  
 if (image != null) {  
 \_infoBarImage.setImageResource(image);  
 }  
 }  
  
  
*//########################  
//## Single line overrides  
//########################* @Override  
 public boolean onCreateOptionsMenu(final Menu menu) {  
 getMenuInflater().inflate(R.menu.*main\_\_menu*, menu);  
 updateSearchFilter("");  
 boolean isCreateTab = \_bottomNav.getSelectedItemId() == R.id.*nav\_mode\_create*;  
 menu.findItem(R.id.*action\_picture\_from\_camera*).setVisible(isCreateTab);  
 menu.findItem(R.id.*action\_picture\_from\_gallery*).setVisible(isCreateTab);  
 menu.findItem(R.id.*action\_search\_meme*).setVisible(isCreateTab);  
  
 \_searchItem = menu.findItem(R.id.*action\_search\_meme*);  
 \_searchView = (SearchView) \_searchItem.getActionView();  
  
 SearchManager searchManager = (SearchManager) getSystemService(*SEARCH\_SERVICE*);  
 \_searchView.setSearchableInfo(searchManager.getSearchableInfo(getComponentName()));  
 \_searchView.setQueryHint(getString(R.string.*search\_meme\_\_appspecific*));  
 if (\_searchView != null) {  
 \_searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 if (query != null) {  
 updateSearchFilter(query);  
 }  
 return false;  
 }**

**Conclusion:**

This capstone project is the biggest and most important project that I worked on during our entire academic career. First, it is not only the final step to getting my bachelor degree, but also it is a great self-capacity-check as I stepped away from what I am used to work with to a new development field. During this period, which was less than 4 months, I got to work on a project that helped me improve my capacities in the computer science field especially that it is a different and new platform using Android. It also added a lot to the knowledge that I have accumulated throughout my academic journey at KIET group of institutions, it widened up my experience in dealing with different platforms, and enhanced my aptitudes to work with similar platforms in real life application which is going to work in our favor in future internships or jobs. Putting the technical part aside, this project had a positive impact on some of my personal traits. It upgraded our punctuality because I had to manage my time to provide the best work I can do in no later time than the deadline by sending our weekly dairies to my supervisor It also helped me develop my team spirit and improved my attitudes regarding listening carefully to others’ opinions and respecting them as I have collaborated with each other which smoothened out the flow of this project. Moreover, having to deal with constant pressure, I learnt to be more efficient and confident with what I am doing, and I managed to prevent stress from getting over me and pushing me to lose control over myself. To conclude, this project was a great experience in my academic career that pushed me on so many levels, technically, professionally, and personally. I am quite satisfied with the outcome that I ended up with, but none of it could have been possible without the help of my supervisor, Dr. Vipin sir, whom I would love to thank extremely.

**REFERENCES**

1. BinstockAndrew(May20,2015). ["Java's20YearsofInnovation"](https://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/). Forbes. [Archiv ed](https://web.archive.org/web/20160314102242/http:/www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/) from the original on March 14, 2016. Retrieved March 18, 2016.
2. ^ [Jump up to:a](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-BarbaraLiskov_2-0) [b](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-BarbaraLiskov_2-1) [c](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-BarbaraLiskov_2-2) [d](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-BarbaraLiskov_2-3) [Barbara Liskov](https://en.wikipedia.org/wiki/Barbara_Liskov) with [John Guttag](https://en.wikipedia.org/wiki/John_Guttag) (2000). Program Development in Java - Abstraction, Specification, and Object-Oriented Design. USA, Addison Wesley. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [9780201657685](https://en.wikipedia.org/wiki/Special:BookSources/9780201657685).
3. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-3) Chaudhary, Harry H. (July 28, 2014). ["Cracking The Java Programming Interview :: 2000+ Java Interview Que/Ans"](https://books.google.com/books?id=0rUtBAAAQBAJ&pg=PAPA133). Retrieved May 29, 2016.
4. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-4) Java 5.0 added several new language features (the [enhanced for loop](https://en.wikipedia.org/wiki/Foreach_loop), [autoboxing](https://en.wikipedia.org/wiki/Object_type_(object-oriented_programming)#Autoboxing), [varargs](https://en.wikipedia.org/wiki/Variadic_function) and [annotations](https://en.wikipedia.org/wiki/Java_annotation)), after they were introduced in the similar (and competing) [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)) language. [[1]](http://www.barrycornelius.com/papers/java5/)[Archived](https://web.archive.org/web/20110319065438/http:/www.barrycornelius.com/papers/java5/) March 19, 2011, at the [Wayback Machine](https://en.wikipedia.org/wiki/Wayback_Machine) [[2]](http://www.levenez.com/lang/)[Archived](https://web.archive.org/web/20060107162045/http:/www.levenez.com/lang/) January 7, 2006, at the [Wayback Machine](https://en.wikipedia.org/wiki/Wayback_Machine)
5. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-5) Gosling, James; McGilton, Henry (May 1996). ["The Java Language Environment"](https://www.oracle.com/technetwork/java/langenv-140151.html). [Archived](https://web.archive.org/web/20140506214653/http:/www.oracle.com/technetwork/java/langenv-140151.html) from the original on May 6, 2014. Retrieved May 6, 2014.
6. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-6) Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad. ["The Java Language Specification, 2nd Edition"](https://java.sun.com/docs/books/jls/second_edition/html/intro.doc.html#237601). [Archived](https://web.archive.org/web/20110805051057/http:/java.sun.com/docs/books/jls/second_edition/html/intro.doc.html#237601) from the original on August 5, 2011. Retrieved February 8, 2008.
7. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-7) ["The A-Z of Programming Languages: Modula-3"](https://web.archive.org/web/20090105145818/http:/www.computerworld.com.au/index.php/id%3B1422447371%3Bpp%3B3%3Bfp%3B4194304%3Bfpid%3B1). Computerworld.com.au. Archived from [the original](http://www.computerworld.com.au/index.php/id;1422447371;pp;3;fp;4194304;fpid;1) on January 5, 2009. Retrieved June 9, 2010*.*
8. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-8) [Niklaus Wirth](https://en.wikipedia.org/wiki/Niklaus_Wirth) stated on a number of public occasions, e.g. in a lecture at the Polytechnic Museum, Moscow in September 2005 (several independent first-hand accounts in Russian exist, e.g. one with an audio recording: *Filippova, Elena (September 22, 2005).*[*"Niklaus Wirth's lecture at the Polytechnic Museum in Moscow"*](http://www.delphikingdom.com/asp/viewitem.asp?catalogid=1155)*.*), that the Sun Java design team licensed the Oberon compiler sources a number of years prior to the release of Java and examined it: a (relative) compactness, type safety, garbage collection, no multiple inheritance for classes – all these key overall design features are shared by Java and Oberon.
9. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-9) [Patrick Naughton](https://en.wikipedia.org/wiki/Patrick_Naughton) cites [Objective-C](https://en.wikipedia.org/wiki/Objective-C) as a strong influence on the design of the Java programming language, stating that notable direct derivatives include Java interfaces (derived from Objective-C's [protocol](https://en.wikipedia.org/wiki/Objective-C#Protocols)) and primitive wrapper classes. [[3]](http://cs.gmu.edu/~sean/stuff/java-objc.html) [Archived](https://web.archive.org/web/20110713014816/http:/cs.gmu.edu/~sean/stuff/java-objc.html) July 13, 2011, at the [Wayback Machine](https://en.wikipedia.org/wiki/Wayback_Machine)
10. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-10) *TechMetrix Research (1999).*[*"History of Java"*](https://web.archive.org/web/20101229090912/http:/www.fscript.org/prof/javapassport.pdf)*(PDF). Java Application Servers Report. Archived from*[*the original*](http://www.fscript.org/prof/javapassport.pdf)*(PDF) on December 29, 2010. The project went ahead under the name greenand the language was based on an old model of*[*UCSD Pascal*](https://en.wikipedia.org/wiki/UCSD_Pascal)*, which makes it possible to generate interpretive code.*
11. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-11) [*"A Conversation with James Gosling – ACM Queue"*](http://queue.acm.org/detail.cfm?id=1017013)*. Queue.acm.org. August 31, 2004.*[*Archived*](https://web.archive.org/web/20150716194245/http:/queue.acm.org/detail.cfm?id=1017013)*from the original on July 16, 2015. Retrieved June 9, 2010.*
12. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-12) In the summer of 1996, Sun was designing the precursor to what is now the event model of the AWT and the JavaBeans component architecture. Borland contributed greatly to this process. We looked very carefully at Delphi Object Pascal and built a working prototype of bound method references in order to understand their interaction with the Java programming language and its APIs.[White Paper About Microsoft's Delegates](https://web.archive.org/web/20120627043929/http:/java.sun.com/docs/white/delegates.html)
13. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-chplspec_13-0) ["Chapel spec (Acknowledgements)"](http://chapel.cray.com/spec/spec-0.98.pdf) (PDF). Cray Inc. October 1, 2015. [Archived](https://web.archive.org/web/20160205114946/http:/chapel.cray.com/spec/spec-0.98.pdf) (PDF) from the original on February 5, 2016. Retrieved January 14, 2016.
14. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-gambas_14-0) ["Gambas Documentation Introduction"](http://gambaswiki.org/wiki/doc/intro?nh&l=en). Gambas Website. [Archived](https://web.archive.org/web/20171009041815/http:/gambaswiki.org/wiki/doc/intro?nh&l=en) from the original on October 9, 2017. Retrieved October 9, 2017.
15. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-15) ["Facebook Q&A: Hack brings static typing to PHP world"](http://www.infoworld.com/article/2610885/facebook-q-a--hack-brings-static-typing-to-php-world.html). InfoWorld. March 26, 2014. [Archived](https://web.archive.org/web/20150213220946/http:/www.infoworld.com/article/2610885/facebook-q-a--hack-brings-static-typing-to-php-world.html) from the original on February 13, 2015. Retrieved January 11, 2015.
16. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-16) ["Write once, run anywhere?"](http://www.computerweekly.com/Articles/2002/05/02/186793/write-once-run-anywhere.htm). [Computer Weekly](https://en.wikipedia.org/wiki/Computer_Weekly). May 2, 2002. Retrieved July 27, 2009.
17. ^ [Jump up to:a](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-design_goals_17-0) [b](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-design_goals_17-1) [c](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-design_goals_17-2) ["1.2 Design Goals of the Java™ Programming Language"](https://www.oracle.com/technetwork/java/intro-141325.html). Oracle. January 1, 1999. [Archived](https://web.archive.org/web/20130123204103/http:/www.oracle.com/technetwork/java/intro-141325.html) from the original on January 23, 2013. Retrieved January 14, 2013.
18. ^ [Jump up to:a](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-:0_18-0) [b](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-:0_18-1) McMillan, Robert (August 1, 2013). ["Is Java Losing Its Mojo?"](https://www.wired.com/2013/01/java-no-longer-a-favorite/). [wired.com](https://en.wikipedia.org/wiki/Wired.com). [Archived](https://web.archive.org/web/20170215115409/https:/www.wired.com/2013/01/java-no-longer-a-favorite/) from the original on February 15, 2017. Retrieved March 8, 2017. Java is on the wane, at least according to one outfit that keeps on eye on the ever-changing world of computer programming languages. For more than a decade, it has dominated the [TIOBE](https://en.wikipedia.org/wiki/TIOBE_index) Programming Community Index, and is back on top – a snapshot of software developer enthusiasm that looks at things like internet search results to measure how much buzz different languages have. But lately, Java has been slipping.
19. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-19) Chan, Rosalie (January 22, 2019). ["The 10 most popular programming languages, according to the 'Facebook for programmers'"](https://www.businessinsider.de/the-10-most-popular-programming-languages-according-to-github-2018-10?op=1). [Business Insider](https://en.wikipedia.org/wiki/Business_Insider). [Archived](https://archive.today/20190629083530/https:/www.businessinsider.com/the-10-most-popular-programming-languages-according-to-github-2018-10?op=1&r=DE&IR=T) from the original on June 29, 2019. Retrieved June 29, 2019.
20. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-20) ["JavaOne 2013 Review: Java Takes on the Internet of Things"](https://www.oracle.com/technetwork/articles/java/afterglow2013-2030343.html). www.oracle.com. [Archived](https://web.archive.org/web/20160419213058/http:/www.oracle.com/technetwork/articles/java/afterglow2013-2030343.html) from the original on April 19, 2016. Retrieved June 19, 2016. [Alt URL](https://www.imarslan.com/javaone-2013-review-java-takes-on-the-internet-of-things)
21. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-21) ["Why should I uninstall older versions of Java from my system?"](https://www.java.com/en/download/faq/remove_olderversions.xml). Oracle. Retrieved September 9, 2016.
22. [^](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_ref-22) Byous, Jon (c. 1998). ["Java technology: The early years"](https://web.archive.org/web/20050420081440/http:/java.sun.com/features/1998/05/birthday.html). Sun Developer Network. [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems). Archived from [the original](https://java.sun.com/features/1998/05/birthday.html)on April 20, 2005. Retrieved April 22, 2005

**BIBLIOGRAPHY**

## **C++ books**

John J. Barton and Lee R. Nackman, *Scientific and Engineering C++*, Addison-Wesley, 1994, ISBN 0-201-53393-6.

This book illustrates the use of C++ to build matrix-like classes to solve scientific and numerical problems. The book shows many advanced idioms that use C++ parameterized types.

Martin D. Carroll and Margaret A. Ellis, *Designing and Coding Reusable C++,* Addison-Wesley, 1995, ISBN 0-201-51284-X.

This book presents many issues in the creation of reusable C++ class libraries: correct use of multiple inheritance, source file organization, error handling, initialization of objects, and other technical C++ topics.

John Lakos, *Large Scale C++ Software Design*, Addison-Wesley, 1996, ISBN 0-201-63362-0.

This book discusses large scale software development issues in C++: how to design software to minimize recompilation and relinking time when changes are made in the middle of a design, building layers of classes to reduce circular dependencies in compilation and testing, and component design principles. Lakos summaries his experiences in using C++ into a well-defined set of design rules, guidelines, and principles.

Scott Meyers, *Effective C++: 50 Specific Ways to Improve Your Programs and Designs*, second edition, Addison-Wesley, 1998, ISBN 0-201-92488-9.

This is the most important intermediate-level C++ book for anyone who plans to do object oriented programming in C++. Scott covers all of the important topics in C++ as a series of 50 short guidelines.

Scott Meyers, *More Effective C++: 35 New Ways to Improve Your Programs and Designs*, Addison-Wesley, 1996, ISBN 0-201-63371-X.

This is an advanced-level follow-on to Scott Meyers’ Effective C++ book. It covers several advanced features of C++, including exception handling, efficiency issues for multiple inheritance and runtime type identification, and abstract classes.

## **Java books**

Ken Arnold and James Gosling, *The Java Programming Language*, Addison-Wesley, 1996, ISBN 0-201-63455-4.

This is the basic introduction to the Java programming language by two of the Java creators at Sun Microsystems.

Gary Cornell and Cay S. Horstmann, *Core Java*, second edition, Prentice-Hall, 1997, ISBN 0-13-766957-7 (vol.1), ISBN 0-13-766965-8 (vol. 2).

These two books are a great introduction to Java for C and C++ programmers. They give excellent examples of good Java usage. This book is much less negative about other languages than most of the books written by people from Sun.

David Flanagan, *Java in a Nutshell*, second edition, O’Reilly & Associates, 1997, ISBN 1-56592-262-X.