Programming: Mobile Applications II

PROG3210

Taso Perdikoulias 2023-2024

Intents

Source: https://developer.android.com/guide/components/intents-filters

Intents

An `Intent` is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental use cases:

- **Starting an activity**

  An `Activity` represents a single screen in an app. You can start a new instance of an `Activity` by passing an `Intent` to `startActivity()`. The `Intent` describes the activity to start and carries any necessary data.

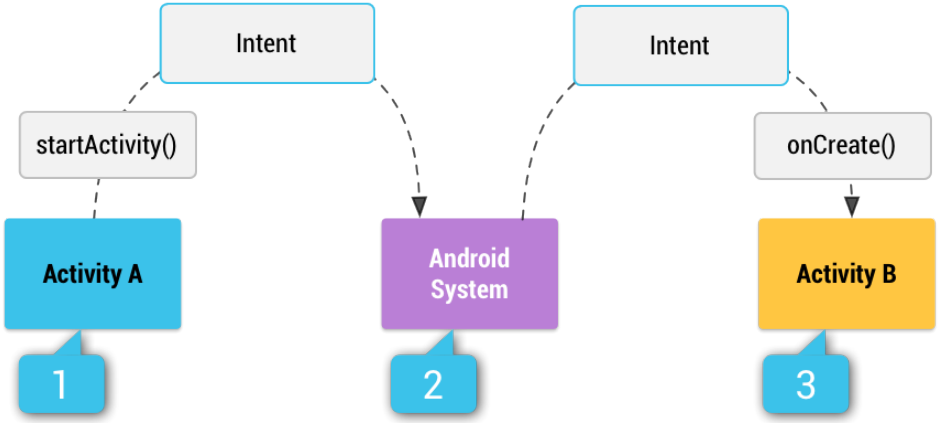
  If you want to receive a result from the activity when it finishes, call `startActivityForResult()`. Your activity receives the result as a separate `Intent` object in your activity's `onActivityResult()` callback. For more information, see the Activities guide.

Source: https://developer.android.com/guide/components/intents-filters

Intent Types

There are two types of intents:

- **Explicit intents** specify which application will satisfy the intent, by supplying either the target app's package name or a fully-qualified component class name. You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, you might start a new activity within your app in response to a user action, or start a service to download a file in the background.
- **Implicit intents** do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

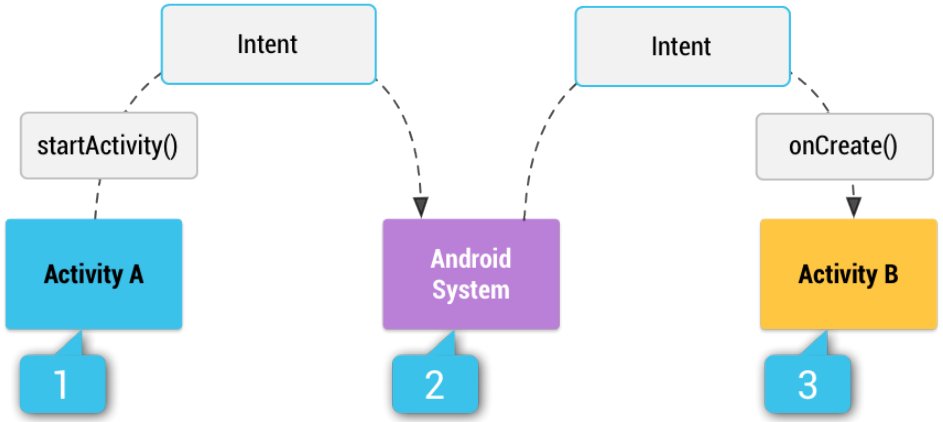


Source: https://developer.android.com/guide/components/intents-filters

Intents

This shows how an intent is used when starting an activity. When the **Intent** object names a specific activity component explicitly, the system immediately starts that component.

When you use an implicit intent, the Android system finds the appropriate component to start by comparing the contents of the intent to the *intent filters* declared in the manifest file of other apps on the device. If the intent matches an intent filter, the system starts that component and delivers it the **intent** object. If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use.



Source: https://developer.android.com/guide/components/intents-filters

Example explicit intent

```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```

Source: https://developer.android.com/guide/components/intents-filters

Example implicit intent

```
// Create the text message with a string.
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Try to invoke the intent.
try {
    startActivity(sendIntent);
} catch (ActivityNotFoundException e) {
    // Define what your app should do if no activity can handle the intent.
}
```

Source: https://developer.android.com/guide/components/intents-filters

Send simple data to other apps

Android uses **intents** and their associated extras to let users share information quickly and easily using their favorite apps.

Android provides two ways for users to share data between apps:

The **Android Sharesheet** is primarily designed for sending content outside your app and/or directly to another user. For example, sharing a URL with a friend.

The Android intent resolver is best suited for passing data to the next stage of a well-defined task.

For example, opening a PDF from your app and letting users pick their preferred viewer.

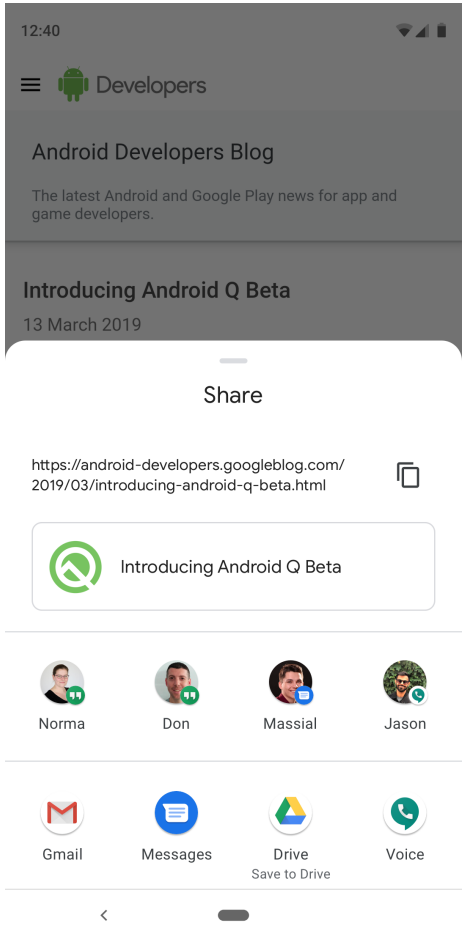
When you construct an intent, you specify the action you want the intent to perform. Android uses the action to send data from one activity to another, even across process boundaries. You need to specify the data and its type. The system automatically identifies the compatible activities that can receive the data and displays them to the user. In the case of the intent resolver, if only one activity can handle the intent, that activity immediately starts.

Source: https://developer.android.com/training/sharing/send#java

Why to use the Android Sharesheet

The Android Sharesheet lets users share information with the right person, with relevant app suggestions, all with a single tap. The Sharesheet can suggest targets unavailable to custom solutions and uses a consistent ranking.

This is because the Sharesheet can take into account information about app and user activity that is only available to the system



Source: https://developer.android.com/training/sharing/send#java

Use the Android Sharesheet

For all types of sharing, create an intent and set its action to Intent.ACTION_SEND. To display the Android Sharesheet, call Intent.createChooser(), passing it your Intent object. It returns a version of your intent that always displays the Android Sharesheet.

Send text content

The most straightforward and common use of the Android Sharesheet is to send text content from one activity to another. For example, most browsers can share the URL of the currently displayed page as text with another app. This is useful for sharing an article or website with friends through email or social networking. Here's an example of how to do this:

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, "This is my text to send.");
sendIntent.setType("text/plain");

Intent shareIntent = Intent.createChooser(sendIntent, null);
startActivity(shareIntent);
```

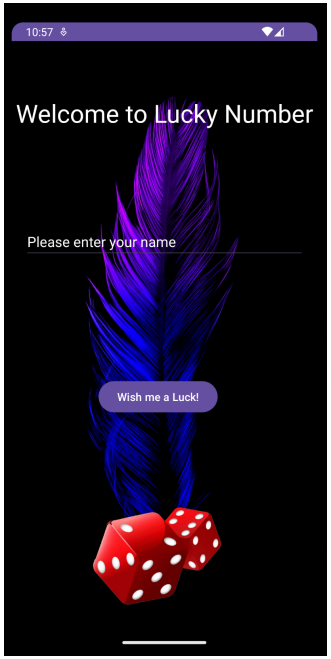
Source: https://developer.android.com/training/sharing/send#java

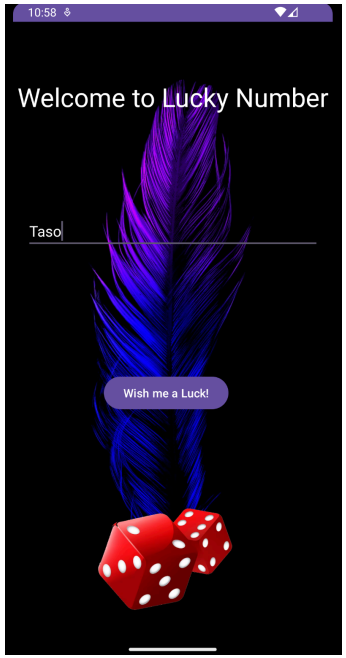
Assignment #1

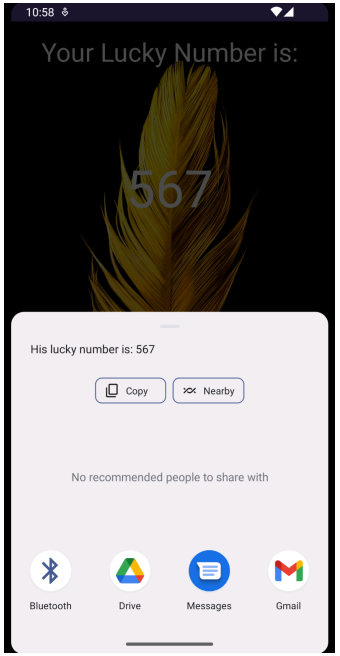
Lucky Number App

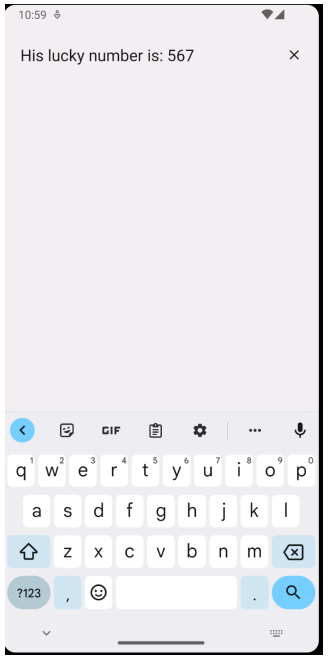
Assignment #1

Fonts/Hint	Background	Second Activity	Share	Paste
Color Size Test Removes when typing	Takes up full background Different for second Activity	Copy and launch second Activity	Share button activates share dialog	Ability to paste shared data to text field (web browser)

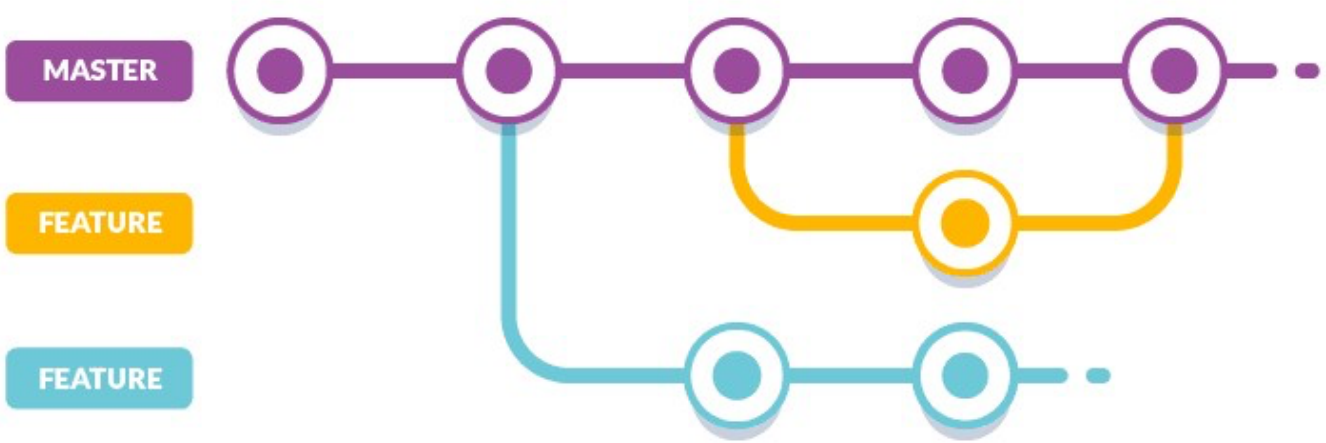








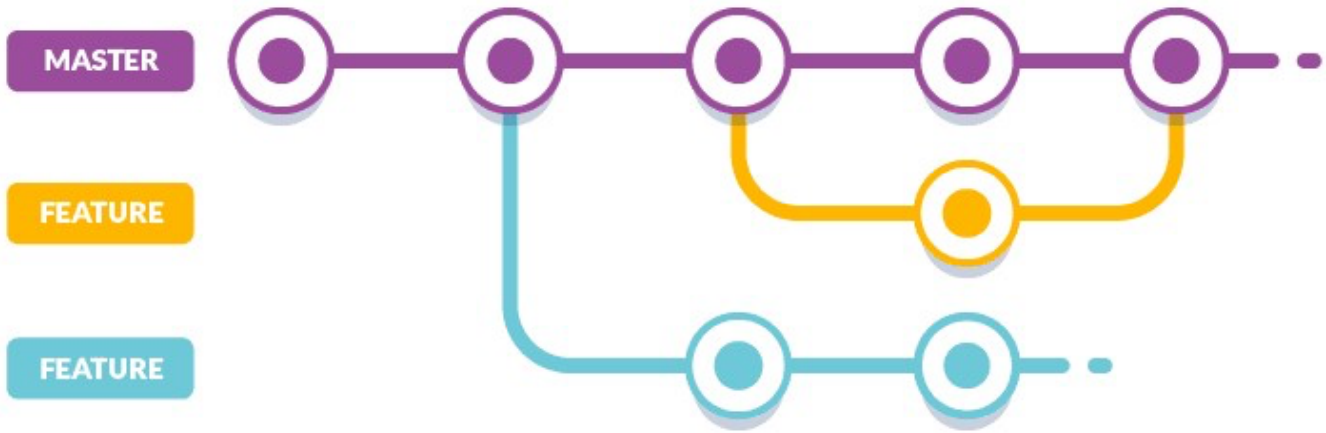
Git and Source Code Repository



Source: https://www.thatcompany.com/wp-content/uploads/2020/03/art5.jpg

Source Code Repository

Source code repositories, also known as version control systems, are essential tools for managing and tracking changes in software development. Here are the core concepts associated with source code repositories



Source: https://www.thatcompany.com/wp-content/uploads/2020/03/art5.jpg

What is git?

Git is a distributed version control system (DVCS) used in software development. It was created by Linus Torvalds in 2005 to address the need for a more efficient and flexible way to manage code in collaborative projects, especially within the Linux kernel development community.

What is GitHub?

GitHub is a web-based platform that provides a wide range of features to enhance collaboration and version control in software development. Some key features of GitHub include:

Repository Hosting:
GitHub allows users to host Git repositories on its servers, providing a centralized location for code storage and collaboration.

Version Control:
GitHub supports Git, a powerful distributed version control system that allows developers to track changes in their codebase.

Collaboration Tools:
Pull Requests: Developers can propose changes to a project and request that they be reviewed and merged into the main codebase.

Issues and Labels:
GitHub provides a robust issue tracking system to manage tasks, bugs, and feature requests. Labels help categorize and prioritize issues.

Comments and Discussions:
Users can comment on code, pull requests, and issues to facilitate communication and collaboration.

Code Review:
GitHub provides a user-friendly interface for reviewing code changes. Reviewers can leave comments, suggest changes, and approve or request revisions to proposed code.

Branching and Merging:
GitHub supports Git's branching model, allowing developers to work on different features or bug fixes in isolated branches and merge them back into the main codebase.

Project Management:
GitHub Projects provides a Kanban-style task board to manage and prioritize work. It helps teams organize tasks, set deadlines, and track progress.

Git vs Github

Key Differences:

Location:
Git is a version control system installed locally on a developer's machine.
GitHub is a web-based platform hosted on GitHub's servers.

Functionality:
Git provides the core version control features (committing, branching, merging) on your local machine.
GitHub adds a web-based interface with features like remote repository hosting, pull requests, issue tracking, and more.

Collaboration:
Git alone is designed for local version control and does not inherently support collaboration.
GitHub facilitates collaboration by providing a platform for developers to share and work on code together.

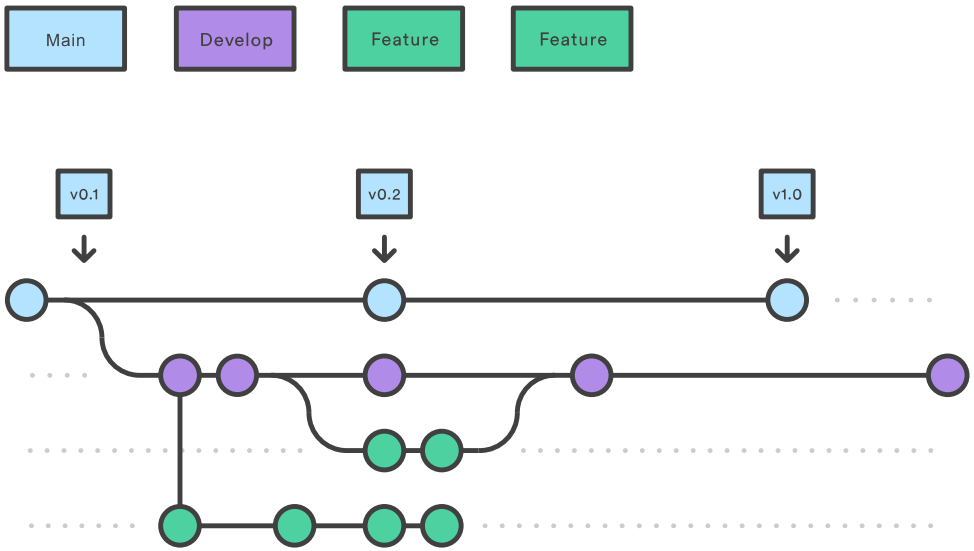
Project Management:
GitHub offers additional features like issue tracking, project boards, and integrations with various development tools, which are not part of Git.

In practice, many developers use Git alongside GitHub to take advantage of both local version control and the collaboration features provided by GitHub. This combination is often referred to as using "Git and GitHub" together.

Gitflow

Gitflow is a branching model for Git that defines a set of branching conventions designed to make collaboration, release management, and versioning more structured and manageable in software development projects.

It was introduced by Vincent Driessen in a blog post and has since become a widely adopted workflow.



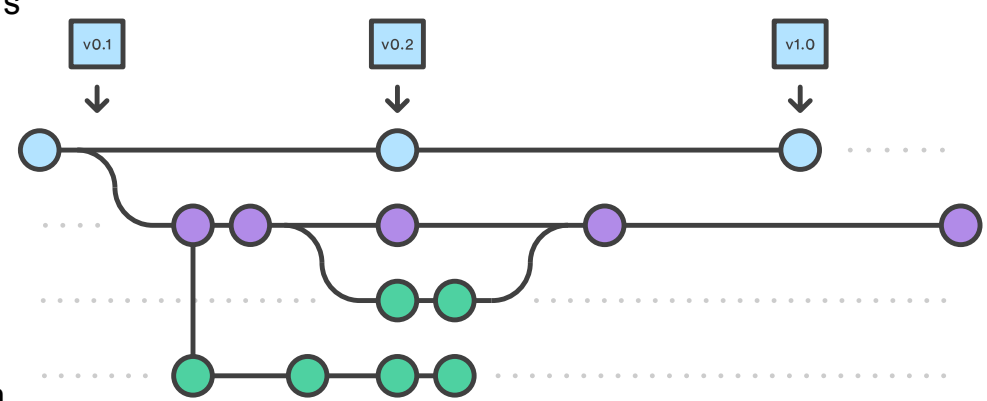
Source: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Gitflow

Here are the key components and concepts of Gitflow:

Main Branches:
Master: The master branch represents the production-ready code. It should always contain the latest stable release of the project.

Develop: The develop branch is where ongoing development occurs. It's considered the "integration" branch where features are merged before they are released.



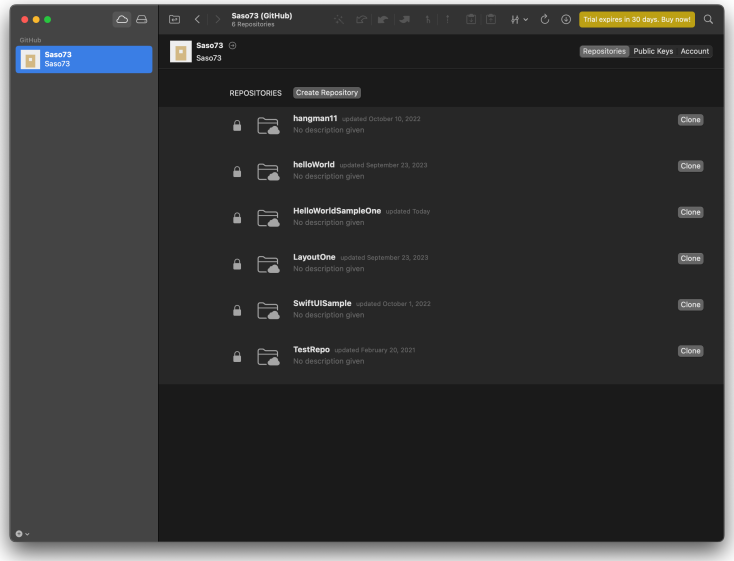
Supporting Branches:
Feature Branches: These branches are used for developing new features or enhancements. They branch off from develop and are merged back into develop when the feature is complete.

Release Branches: Release branches are used to prepare the codebase for a new release. They are created from the develop branch and serve as a stabilization area for last-minute bug fixes or preparation tasks. Once the release is deemed stable, it's merged into both master and develop.

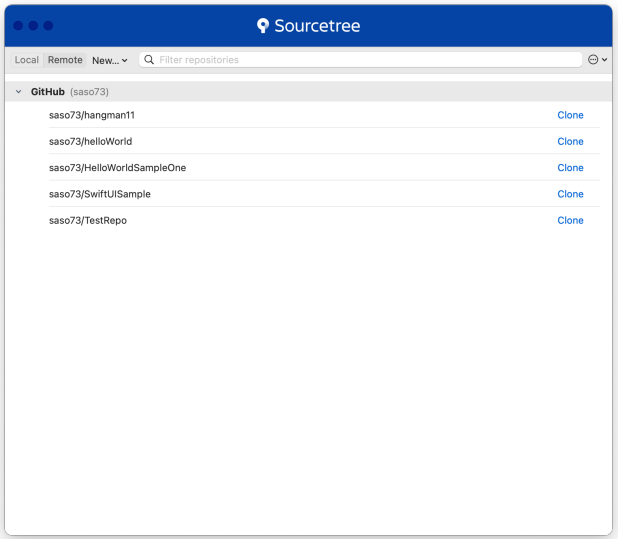
Hotfix Branches: Hotfix branches are used for fixing critical issues in the production code. They branch off from master, and once

Source: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

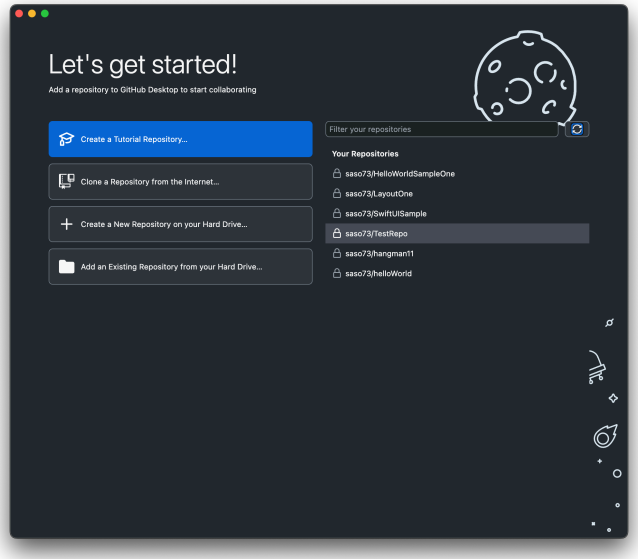
Tools



Tower



Sourcetree



Github Desktop

Exercise

To Do:

Create a Github account
Go to website and create and login to your account

Github Desktop:
Download and install GitHub desktop
Connect it to your Github Account

Create and Upload a HelloWorld Android Application
Create a local and remote repo

Branching:
Create three branches (master, develop, stage)

Collaborator:
Invite at least collaborator to commit 1 change to your Hello World Application
Have them create a feature branch and a “pull request”