REFACTORING

Slides based on slides from previous years.

What is Refactoring?

- Refactoring (noun): a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.
- Refactor (verb): to restructure software by applying a series of refactorings without changing its observable behavior.

Examples of Refactoring

- Encapsulate a field
- Rename a class or method
- Create a constant
- Convert a local variable to field
- Extract a method
- Extract a local variable

Why Refactor?

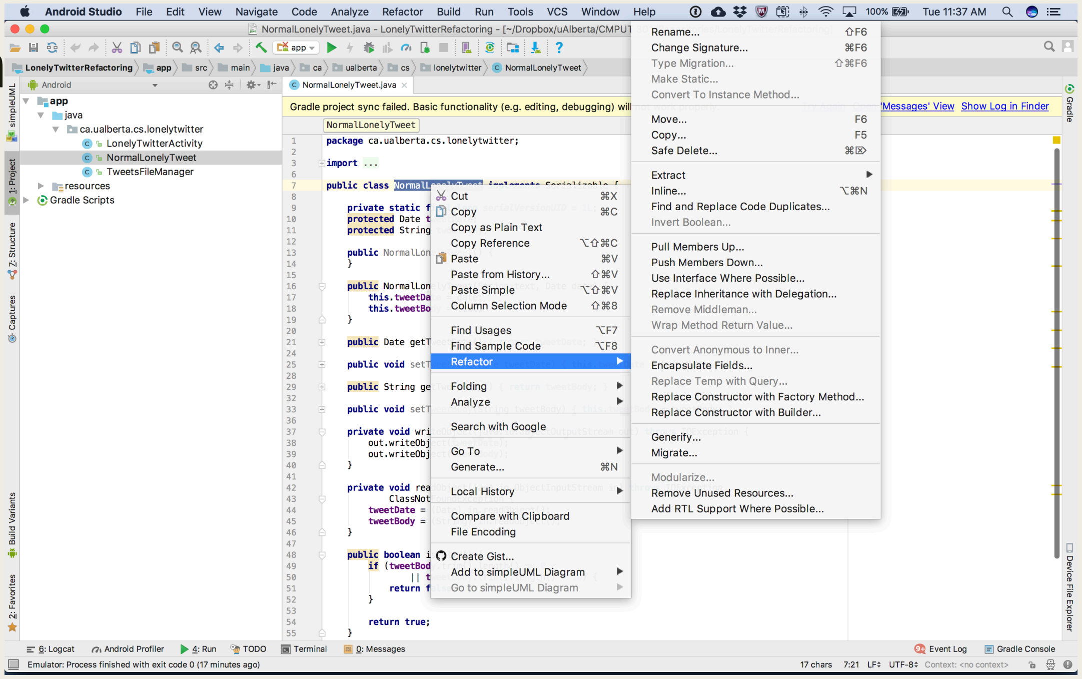
- Improve and evolve the design of a program
- Make software easier to understand and maintain
- Find bugs by clarifying the structure of a program
- Improve programming efficiency

Auto-Refactoring

- Built-in refactoring tools in IDEs like Android Studio, Eclipse, etc.
- Auto-refactoring allows you to update design of your code without breaking it.
- Reduces the effort to fix a poorly coded program.
- Testing still may be necessary.

How to Use Refactor?

- Select the element in the source code
- Right click of the element you want to refactor
- Go to the Refactor menu



Gradle project sync failed. Basic functionality (e.g. editing, debugging) will

```
package ca.ualberta.cs.lonelytwitter;

import ...

public class NormalLonelyTweet implements Serializable {

    private static f
    protected Date t
    protected String

    public NormalLon
    }

    public NormalLon
        this.tweetDa
        this.tweetBo
    }

    public Date getT

    public void setT

    public String ge

    public void setT

    private void wri
        out.writeObj
        out.writeObj
    }

    private void rea
            ClassNot
        tweetDate =
        tweetBody =
    }

    public boolean i
        if (tweetBod
                || t
            return f
        }

        return true;
    }
```

Emulator: Process finished with exit code 0 (17 minutes ago)

Common Refactoring Actions

- Rename
- Move
- Change method signature
- Extract method
- Extract local variable
- Extract constant
- Convert local variable to field
- Extract superclass
- Extract interface
- Push down
- Pull up
- Introduce parameter

When to Refactor?

- Add functionality
- When performing code reviews
- “Rule of three: The first time you do something, you just do it. The second time you do something similar, you wince at the duplication, but you do the duplicate thing anyway. The third time you do something similar, you refactor.” **Rule of Three. Don Roberts**

In-class Exercise: Setup

- Fork the following project: alisajedi/LonelyTwitterRefactoring
- Clone <u>YOUR</u> LonelyTwitterRefactoring project
- Open the project in Android Studio

In-Class Exercise: Task 1

NormalLonelyTweet:

- Allows up to 10 characters for the tweet
- We want to have a similar class that: 1) allows string of length 20 and 2) shows the string in uppercase.

Do steps of approach A or B:

Approach A

- *Make a superclass (LonelyTweet) and move most methods from NormalLonelyTweet into that*
- *Duplicate NormalLonelyTweet as ImportantLonelyTweet*

Approach B

- *Rename NormalLonelyTweet to LonelyTweet*
- *Create two new subclasses of LonelyTweet: 1) NormalLonelyTweet and 2) ImportantLonelyTweet*

In-Class Exercise: Task 2

- LonelyTweet: getTweetBody()
- We need two different implementations for the two subclasses (one returns normal string , and the other returns uppercase)
- Push down the method to the two sub-classes and change the appropriate one.

In-Class Exercise: Task 3

- TweetFileManager: extract a constant for the file name

In-Class Exercise: Task 4

- LonelyTwitterActivity: we need getters and setters for “tweets”

In-Class Exercise: Task 5

- Move the superclass and its sub-classes to a new package (lonelytweet)

Lab Exercise

- Use Analyze -> Inspect code on your lonelyTwitter repository.
- Find and fix 5 different issues that are mentioned by the code inspection. Mark your fixes with a small comment saying what you fixed and why. Submit a link to your modified version of lonelyTwitter.

---OR---

- Get into your project groups, run the code inspection tool on your project and find 5 issues that should be fixed. Write a small description of each issue and why it should be addressed. Everyone in the group should submit.