Assignment 2: Serving Size Calculator

Submit a ZIP file or your code and the GitLab URL to CourSys: [https://courses.cs.sfu.ca/](https://courses.cs.sfu.ca/)

Late penalty is 10% per calendar day (each 0 to 24 hour period past due).

Maximum submission 2 days late.

This assignment is to be done **individually or in pairs**. Do not share your code or solution, do not copy code found online, and do not post questions about the assignment online. Ask questions on Piazza discussion forum (see webpage).

You may use code provide by the instructor, or in help guides/videos provided by the instructor (such as my YouTube videos).

You may follow any guides you like online as long as it is providing you information on how to solve the problem vs a solution to the assignment.

If receiving help from someone, they must be teaching you not writing your code.

You may not resubmit code you have previously submitted for any course offering.

1. **App Overview**

   1. User inputs one or more pots’ name and weight into app.
   2. In real life: user then uses a pot to make a meal and weighs the pot with the food in it.
   3. In app: user selects the pot they used and enters the combined weight of the pot plus the food. Tool subtracts pot’s weight (when empty) from the total weight to compute weight of just the food.
   4. Finally, user enters how many servings are in the food and tool divides the weight of food by number of servings, displaying weight of each serving.

2. **Pot Class JUnit**

   - Using the skeleton online, implement the Pot class.
   - Write JUnit tests for your Pot class.
     - You must achieve 100% code coverage of Pot’s lines code.
     - You must write at least 4 test methods (may be closer to 8+).
     - All tests must pass, and be well named.

3. **Required Application Features**

   Implementing “**Required Features**” can earn a **good grade**; to get a **great grade**, you must also complete some “**Optional**” features (listed later). See marking guide for details.

3.1 **General Requirements**

   - Create an Android application targeting the minimum SDK version API 24 or lower.
   - Change the name of your application (shown on all activities). Do this in `strings.xml`.
   - All layout XML files may not have any hard-coded text displayed on the UI: it must come from `strings.xml`. Java code may still hard-code strings/outputs.
   - Screen-shots in this document are for inspiration; as long as your application correctly implements the required features, any nice and usable UI is fine.
   - None of the things listed as “hints” are required; you may choose to do them or not.
   - Activity files (.java and .xml) must be well named, but need not match this document.
   - Create a GitLab repo on gitlab.cs.sfu.ca.
     Commit your changes often (at least every 4 hours of work).
3.2 Screen 1: Pot List

- This is the initial activity displayed when app starts.
- Display the pots which the user has already entered.
  - Show each pot’s name and weight (in grams).
- Use the PotCollection class (provided on assignments webpage) to store the pots.
  - Use the Pot class to store an individual pot.
- Allow the user to add a new pot to the collection by launching the Add Pot activity.
- User may tap on a pot in the list to launch the Calculate activity.

Hints

- Instantiate a PotCollection as a member field of your Pot List activity class.
  - You may edit the PotCollection class as needed.
  - The PotCollection is the model: it contains the data for your application.
- Use a ListView to show the list of pots.
  - Populate the ListView using an ArrayAdapter. See video: https://www.youtube.com/watch?v=eAPFgC9URqc
  - PotCollection class has the getPotDescriptions() method which can give you the array of Strings needed to populate the ListView via an ArrayAdapter.
  - For initial testing (before you have any other screens working), try putting some test data into yourPotCollection object.
- After adding a new pot you’ll need to refresh your list of pots. Just create a new ArrayAdapter and call setAdapter() on the ListView.
  - Put this code in its own method to avoid duplicate code.
  - For how to pass data to and from other activities, see hints for the next two activities.
3.3 Screen 2: Add Pot

- Have entry boxes for the pot’s name and pot’s weight.
  - If using EditText widgets for data input, each must have a hint for what goes in it (such as show in UI to the right).
  - May use other data entry widgets if desired.
  - The weight entry box must only allow non-negative integer values.
  - Have a way of either accepting (OK), or canceling adding the pot (such as the OK / Cancel buttons).

Hints

- Use a table layout for having the two EditText views line up.
- Likely use TextEdit boxes for data entry.
- Convert a String to an int with:
  ```java
  int val = Integer.parseInt("42");
  ```
  - Note that if the String does not contain a number it throws NumberFormatException.
  - You need-not (for the required features) handle validating the user’s inputs (see Optional Features section). So if user does not enter a weight and clicks OK, your application may crash; this is fine (for the core features!).

Requirement on Passing Data back from an activity

- Data is passed from one activity to another using an Intent. See video: https://www.youtube.com/watch?v=SaXYFHJYGLj4
  - Watch the video! It shows the required way of creating an intent and respecting encapsulation.
- Have the Add Pot activity “return” an Intent which stores the following in its Extra:
  - pot name (string)
  - pot weight in g (int)
- Call this from List Pot’s onActivityResult(..) method.
- Process
  - Pot List activity calls startActivityForResult(...) to launch this activity.
  - Pot List activity overrides onActivityResult(...) to handle the “returned” data.
  - Rather than having Pot List activity know how to handle the “returned” intent from the Add Pot activity, have the Add Pot activity expose the following method:
    ```java
    public static Pot getPotFromIntent(Intent data);
    ```
  - This encapsulates the details of how data is stored in the Intent to inside this activity.
  - Call this method and pass in the “returned” intent and it extracts the data about the pot and puts it into a Pot object for us.
3.4 Screen 3: Calculate Serving

- Display the selected pot’s name and weight.
- Allow user to enter the combined weight of the pot with food in it.
  - UI must allow only a non-negative integer.
- Calculate and display the weight of the food (in g)
- Allow user to enter number of servings
  - Must be a non-negative integer.
- Calculate and display the weight of each serving (truncated or rounded to the nearest gram is fine).
- Your activity must not crash if the user has not yet entered a number of servings, or if the number of servings is zero.
- Have some mechanism to navigate back to the previous activity.
- As the user types data into, or changes a value in either of the input boxes, recalculate the serving weight.
- Like the Add Pot activity, you must use `public static` method(s) to encapsulate creating/processing the Intent to inside this class; see hint for suggestions on how.

Hints

- When clicking your “Back” button (or the like), call the `finish()` method to end the activity.
- Pass data to the Calculate activity using an Intent.
  - Pass in the pot name and weight.
  - For good encapsulation, have the Calculate activity expose a method which creates the intent required to launch it. Pass this function the data to be encoded into the Intent. This encapsulates how the pot’s information is stored in the Intent’s extras:
    ```java
    public static Intent makeLaunchIntent(Context context, Pot pot);
    
    To recompute the serving-weight while the user is entering data, you’ll need to pass a
    `TextWatcher` object to the `addTextChangedListener(…)` method of the
    EditText. In this TextWatcher, put your code in `afterTextChanged(…)` which
    recomputes the serving weight; other methods in TextWatcher can be left untouched.
- If trying to display a number in a `TextView`, note that
  ```java
  myTextView.setText(42);
  
  will attempt to load the `strings.xml` resource which has number 42 into the `TextView`,
  which likely does not exist and will crash your program. Instead, convert the into to a
  String:
  ```java
  myTextView.setText("" + 42);
  ```
4. Optional Features

- By completing one or more of these features, you stand to move from “Good work” to “Great work”. See marking guide.
- You may only get marks for the optional features if the required parts of the application work well.
- If you attempt any of these features, your application’s first screen (Pot List activity) must state the features you added. For example, have a TextView at the bottom of the screen listing the features/directions.
- Each feature must have a reasonable user interface.

4.1 Edit Pot

- Support editing a pot stored in the list of pots by long-pressing on the pot.
- Hints:
  - Re-use the Add Pot activity and just pass it extra data (via an Intent). This extra data is the current name and weight of the pot.
  - Have the Pot List activity track which pot is being edited and then when it gets back the new pot data (via the Intent), have it change the existing pot in the PotCollection instead of adding a new one.

4.2 Delete Pot

- Support removing a pot from the list of pots.
- Your choice how to do this; UI must be clear to user.

4.3 Error Checking Input

- Error check all user input.
- Enforce at least the following:
  - Pot name must be at least one character long.
  - Pot weight must be greater than or equal to zero.
- When you detect an error, put up a toast with a helpful error message.

4.4 Save Data

- Save between executions of your application all the pot data the user entered.
- You may want to use SharedPreferences.
- You may want to edit the PotCollection object to support working with a SharedPreference.
- You may use external serialization libraries if you like (Gson/JSON/....).

4.5 Action Bar Buttons

- Use the action bar at the top of the activities to give the following buttons:
  - Pot List activity: Add a pot
  - Add Pot activity: OK and Cancel
  - Calculate activity: Back to Pot List activity.
- When adding these buttons to the action bar, you may remove any duplicate buttons from the rest of the user interface.
4.6 Pot Icons

- Significantly enhance the user interface by allowing the user to set an icon or images for each pot.
- Change the Pot List display to feature a complex layout with the pot’s icon/image and text.
- See video https://www.youtube.com/watch?v=WRANgDgM2Zg

5. Deliverables

To CourSys (https://courses.cs.sfu.ca/) you must submit:

1. Zip file of your project folder, as shown in this tutorial video:
   https://youtu.be/238A85m45-w

2. URL and Tag for your Git repository:
   1. Add the TA for the course as a “Developer” member of your repo:
      - Goto csil-git1.cs.surrey.sfu.ca and select your project
      - On the left hand side, click the cog-wheel drop-down (‘Settings’)
      - Select “Members”
      - Add the TA to your repo as a Developer. You can find the TA’s SFU ID (email address before the @sfu.ca) on the course website’s Course Info page.

   2. Create a tag for your submission as follows:
      - In Android Studio, go to VCS --> Git --> Tag...
      - Enter a name for your tag, such as: final_submission
      - Leave Commit and Message blank.
      - Click Create Tag
      - Push changes to remote repo. On “Push Commits” dialog, select “Push Tags: All”.
      - You can check the tag was pushed correctly in GitLab online.
      - (If you resubmit, create a new tag as above and submit the new tag via CourSys).

3. Submit the git@... URL and tag name to CourSys
   - Find Git URL on csil-git1.cs.surrey.sfu.ca/. Should be similar to:
     git@csil-git1.cs.surrey.sfu.ca:yourid/myProjName.git
   - The “tag” is the name you used above, such as “final_submission”

Please remember that all submissions will automatically be compared for unexplainable similar submissions. Everyone’s submissions will be quite similar, given the nature of this assignment, but please make sure you do your own original work; we will still be checking.