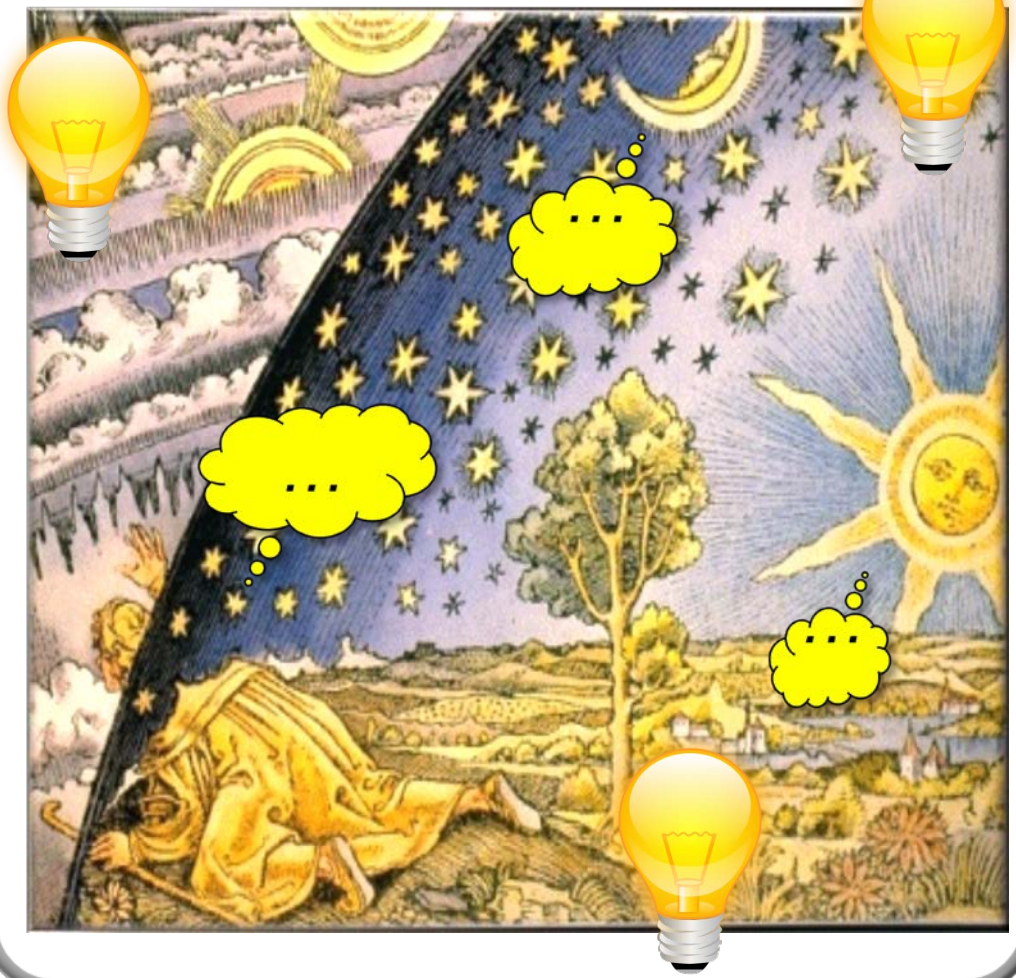
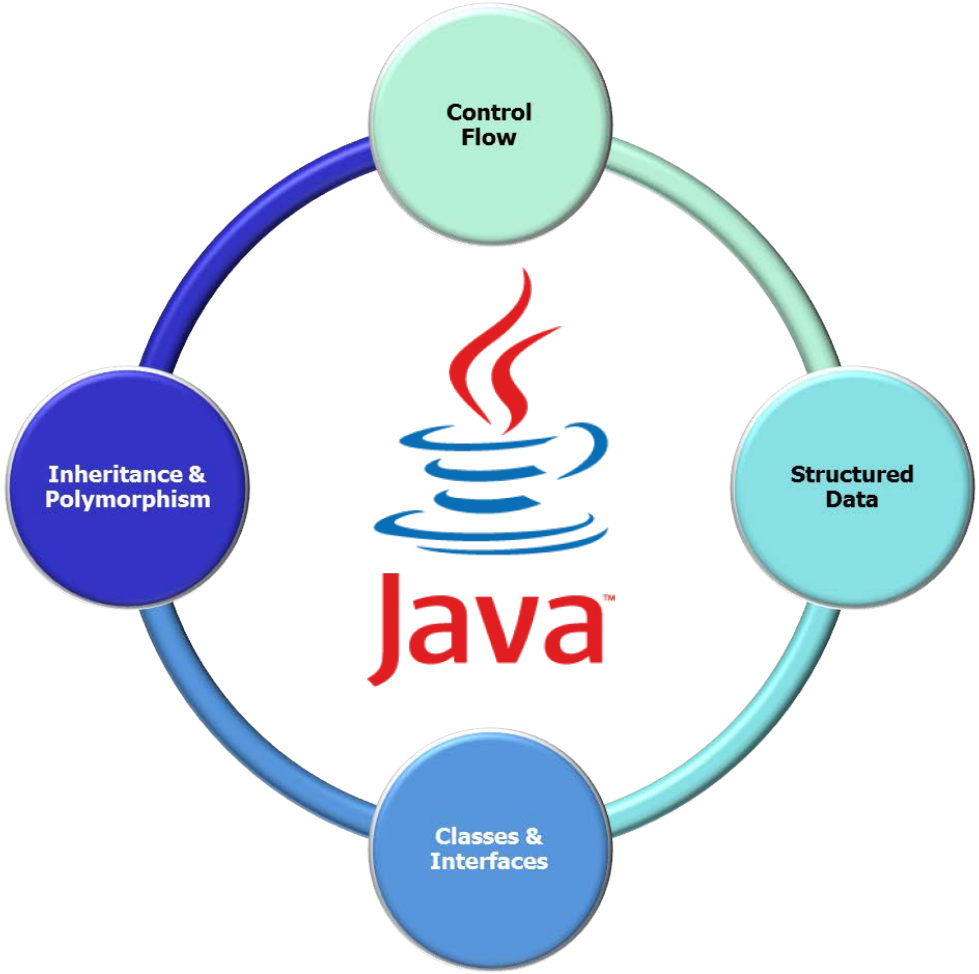
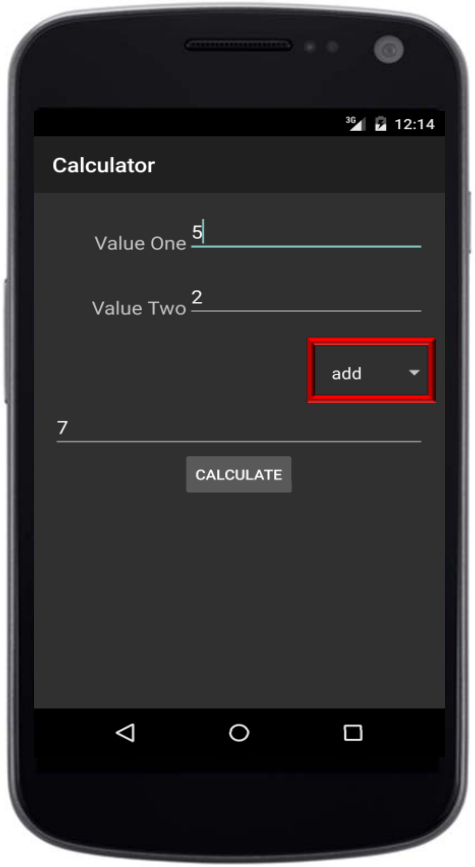
Java for Android: Calculator App



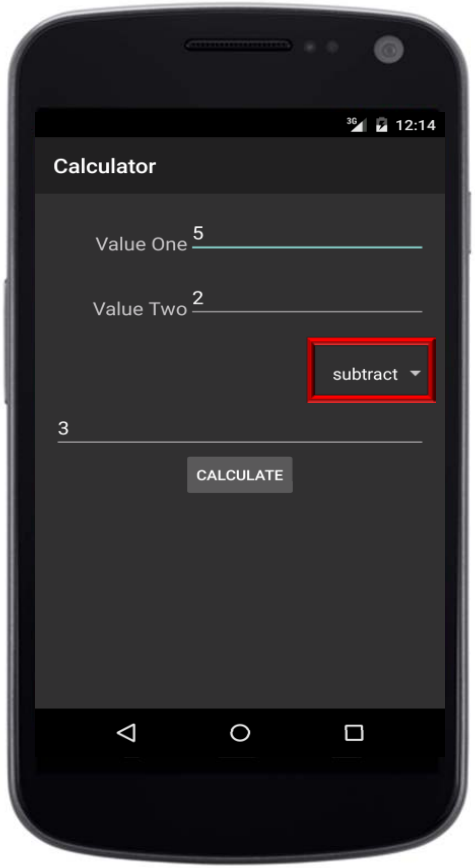
Java for Android: Calculator App



Java for Android: Calculator App



Java for Android: Calculator App



Java for Android: Calculator App

Java for Android: Calculator App

Java for Android: Calculator App

Learning objectives:

- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

Java for Android: Calculator App

Learning objectives:

- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

Java for Android: Calculator App

Learning objectives:

- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

Java for Android: Calculator App

Learning objectives:

- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

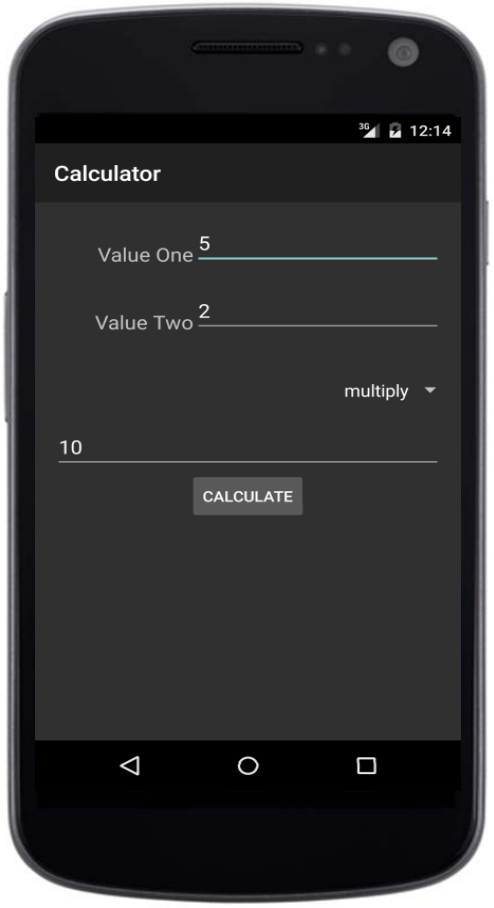
Java for Android: Calculator App

Learning objectives:

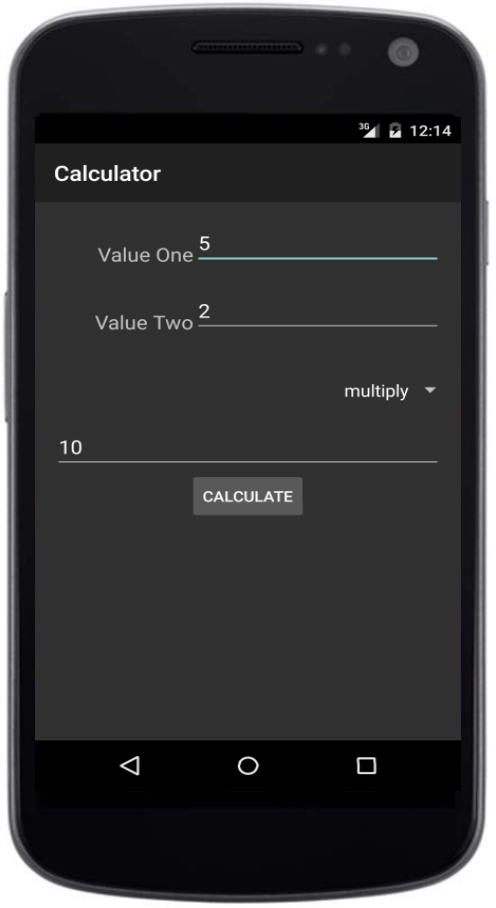
- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

Java for Android: Calculator App

Java for Android: Calculator App



Java for Android: Calculator App



Mini-Project Overview & Requirements

Mini-Project Overview & Requirements

- In this mini-project you'll write the program logic needed to complete a simple calculator app in Java

Mini-Project Overview & Requirements

- In this mini-project you'll write the program logic needed to complete a simple calculator app in Java
  - Performs integer arithmetic on values entered via Android's user interface (UI)

Mini-Project Overview & Requirements

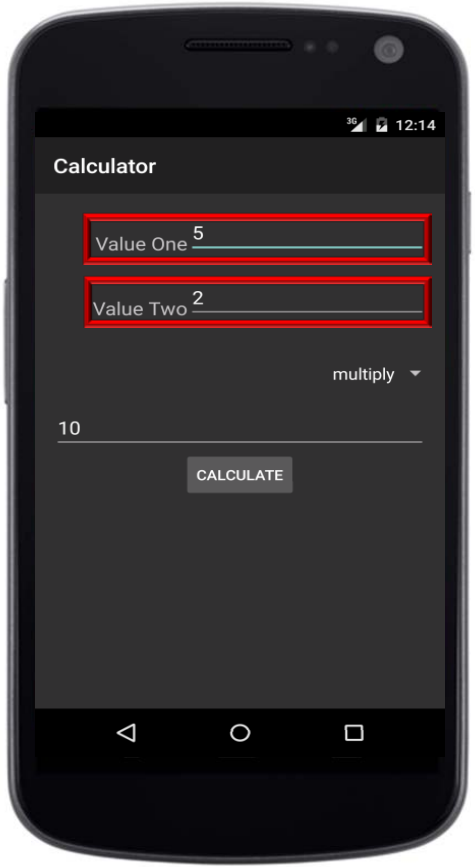
- In this mini-project you'll write the program logic needed to complete a simple calculator app in Java
  - Performs integer arithmetic on values entered via Android's user interface (UI)
  - We supply you skeleton code that implements the app's UI in Android

Mini-Project Overview & Requirements

- Your app should meet several requirements

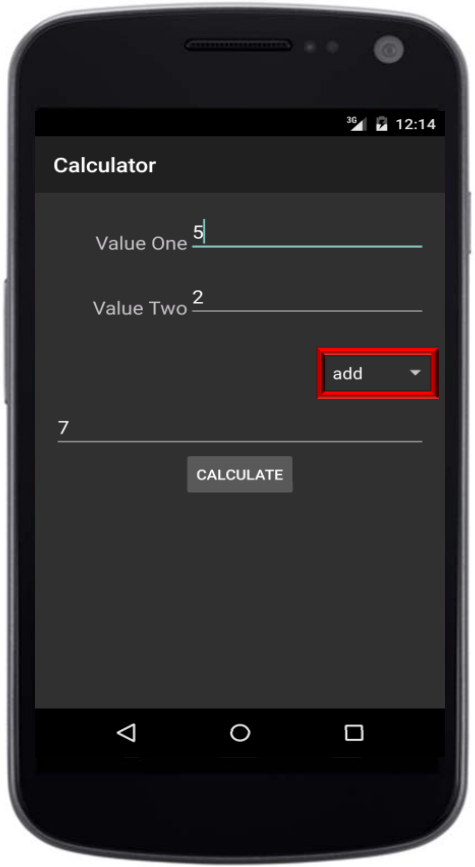
Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations



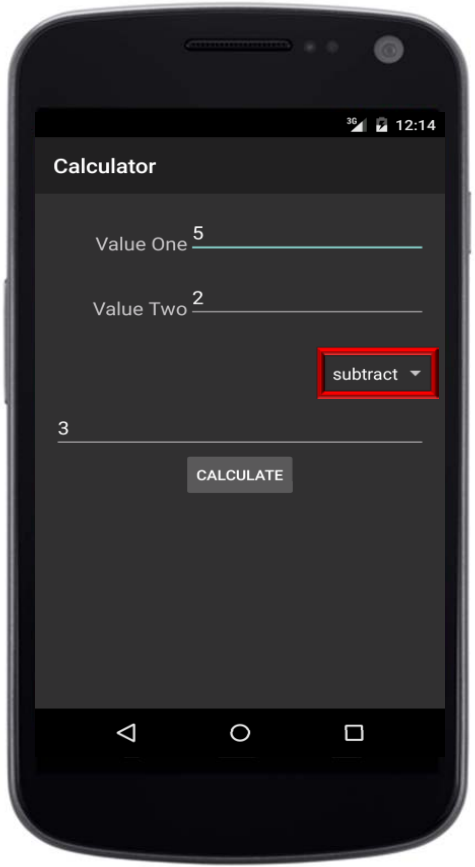
Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
    - Addition



Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
    - Addition
    - Subtraction



Mini-Project Overview & Requirements

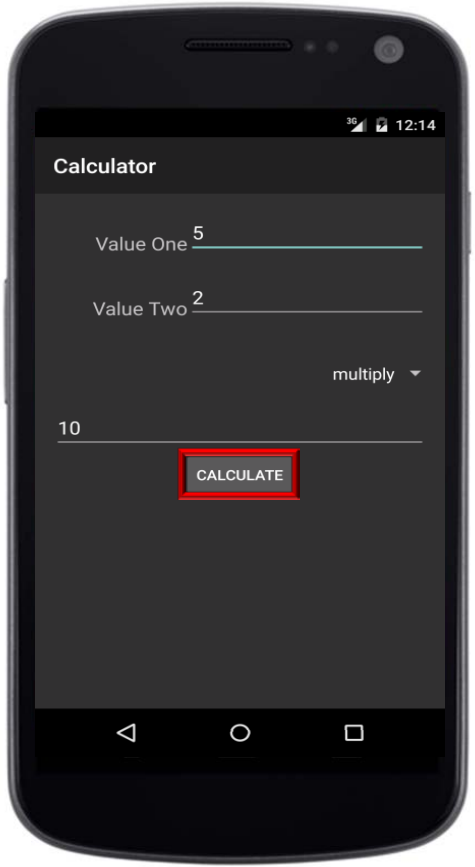
- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
    - Addition
    - Subtraction
    - Multiplication

Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
    - Addition
    - Subtraction
    - Multiplication
    - Division

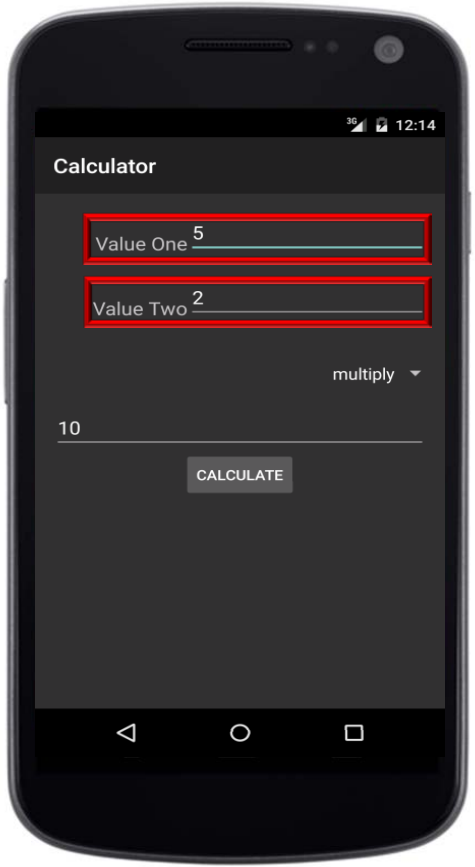
Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
  - After supplying integer values & pressing “calculate”, 3 entities will be provided



Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
  - After supplying integer values & pressing “calculate”, 3 entities will be provided
    - the two integer values entered by the user &

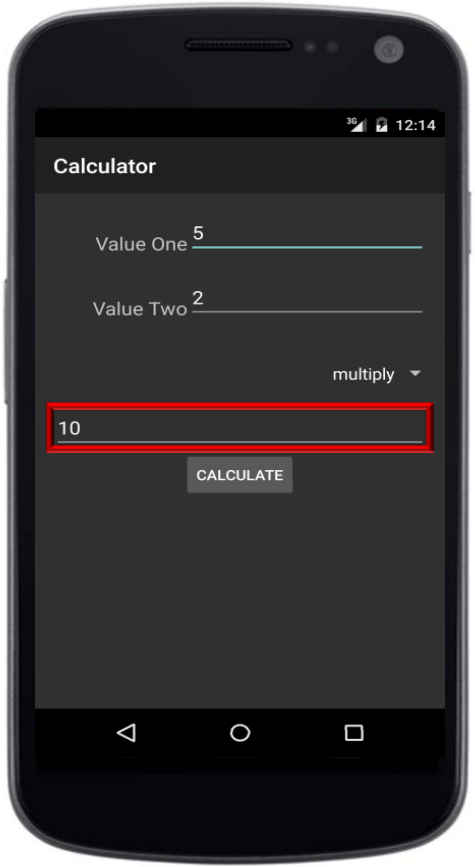


Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
  - After supplying integer values & pressing “calculate”, 3 entities will be provided
    - the two integer values entered by the user &
    - the operation they selected to perform on these values

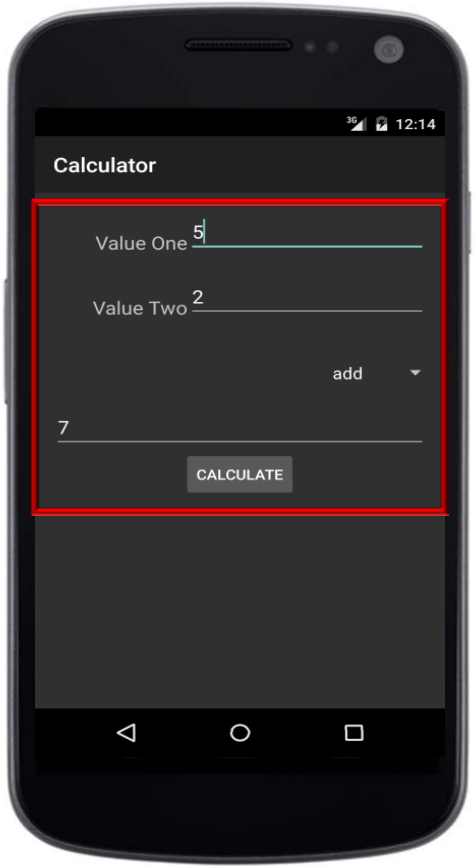
Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
  - After supplying integer values & pressing “calculate”, 3 entities will be provided
  - Your code must perform computation & print a string containing the final answer



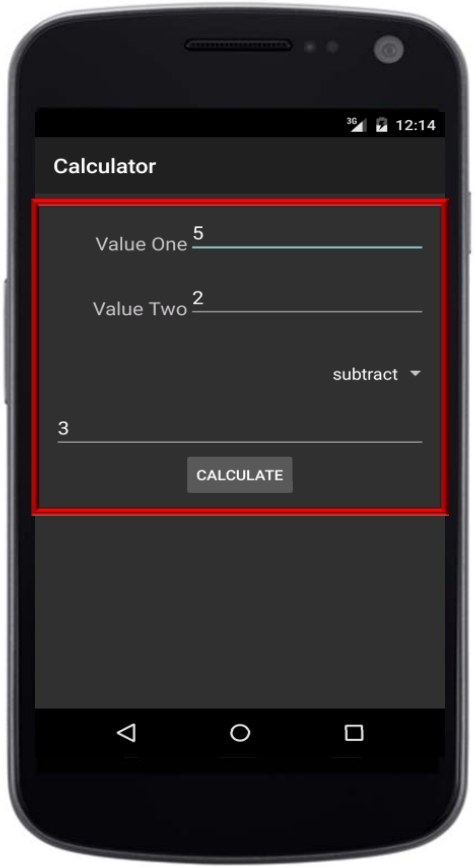
Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
  - After supplying integer values & pressing “calculate”, 3 entities will be provided
  - Your code must perform computation & print a string containing the final answer
  - Results for integer addition, subtraction, & multiplication are what you’d expect



Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
  - After supplying integer values & pressing “calculate”, 3 entities will be provided
  - Your code must perform computation & print a string containing the final answer
  - Results for integer addition, subtraction, & multiplication are what you’d expect



Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
  - After supplying integer values & pressing “calculate”, 3 entities will be provided
  - Your code must perform computation & print a string containing the final answer
  - Results for integer addition, subtraction, & multiplication are what you’d expect

Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
  - After supplying integer values & pressing “calculate”, 3 entities will be provided
  - Your code must perform computation & print a string containing the final answer
  - Results for integer addition, subtraction, & multiplication are what you’d expect
  - Results for integer division must include both the quotient & the remainder (even if the remainder is zero)

Mini-Project Overview & Requirements

- Your app should meet several requirements
  - The provided UI allows user to enter two integer values & select one of 4 operations
  - After supplying integer values & pressing “calculate”, 3 entities will be provided
  - Your code must perform computation & print a string containing the final answer
  - Results for integer addition, subtraction, & multiplication are what you’d expect
  - Results for integer division must include both the quotient & the remainder (even if the remainder is zero)

Steps for Getting Started

Steps for Getting Started

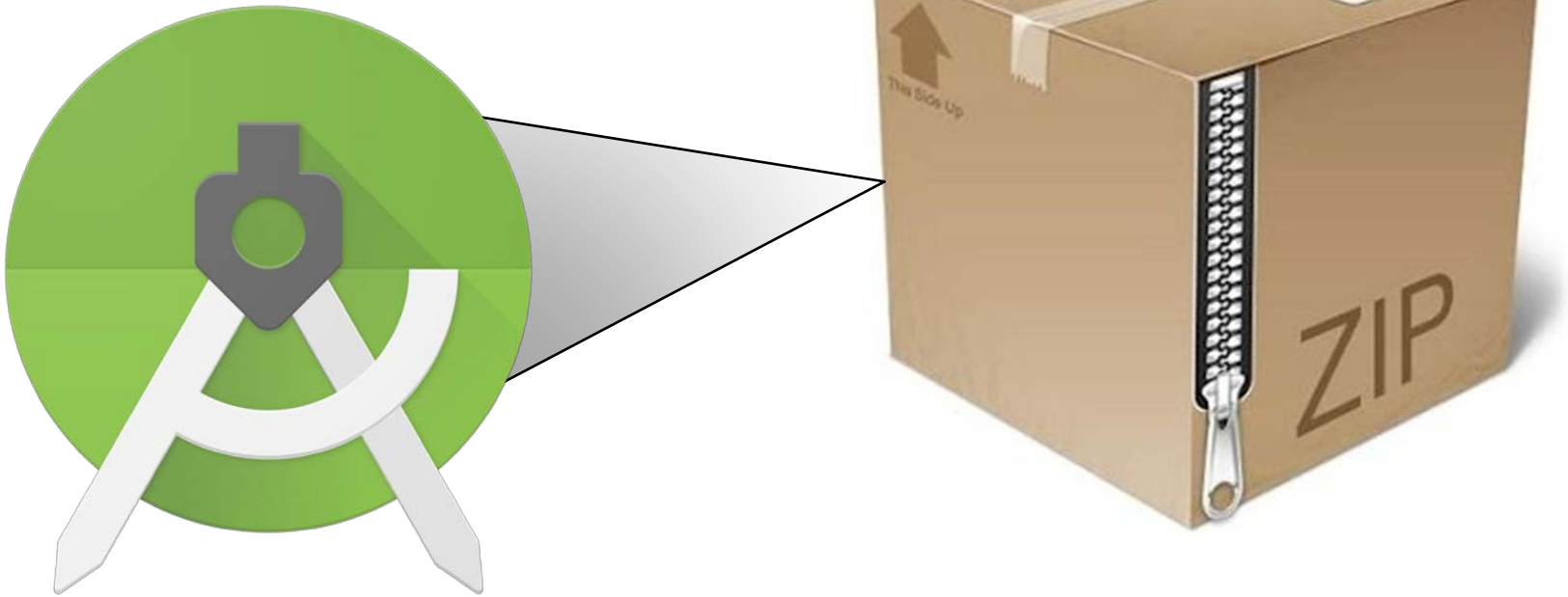
- Start by downloading the supplied zip file & extract the contents onto your computer

Steps for Getting Started

- Start by downloading the supplied zip file & extract the contents onto your computer

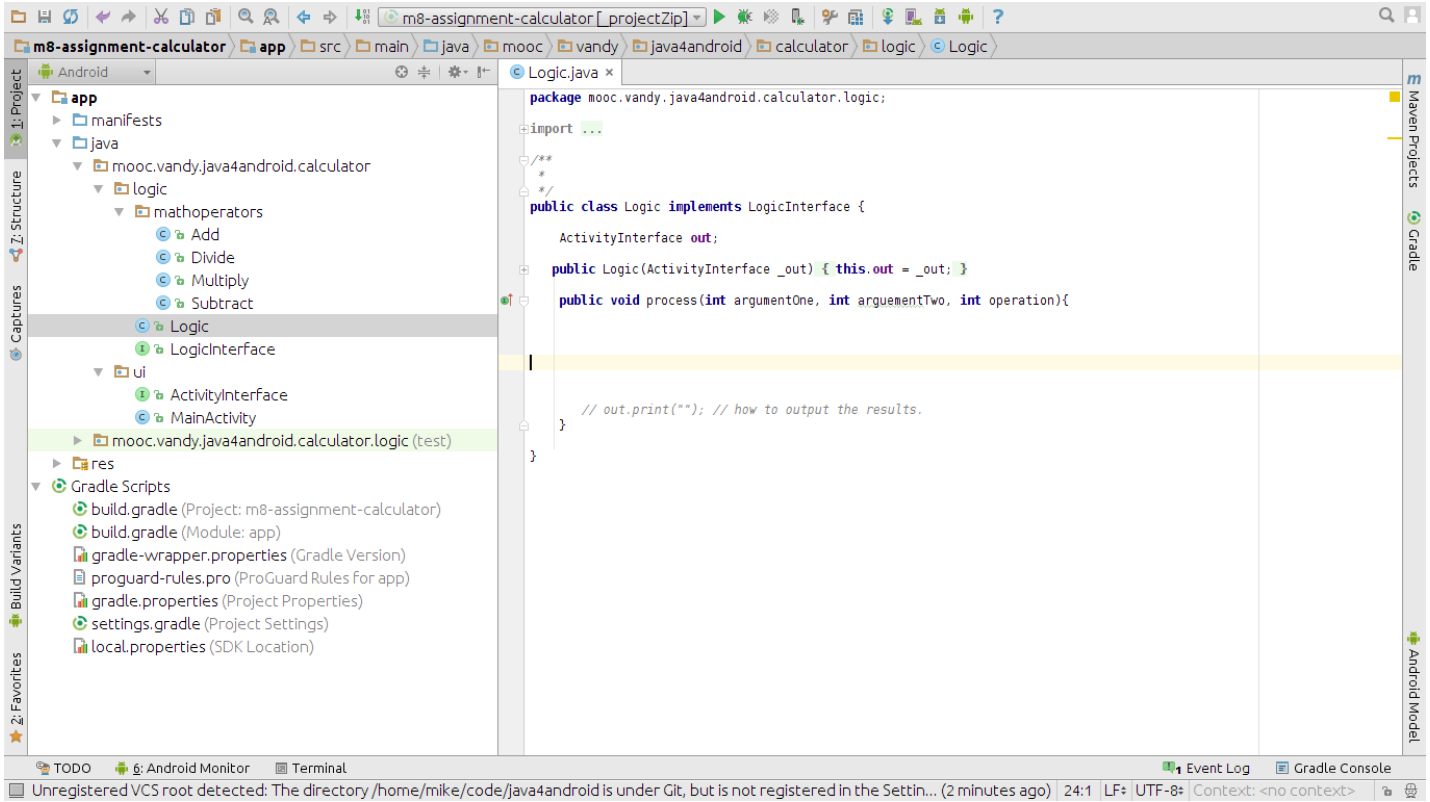
Steps for Getting Started

- Start by downloading the supplied zip file & extract the contents onto your computer



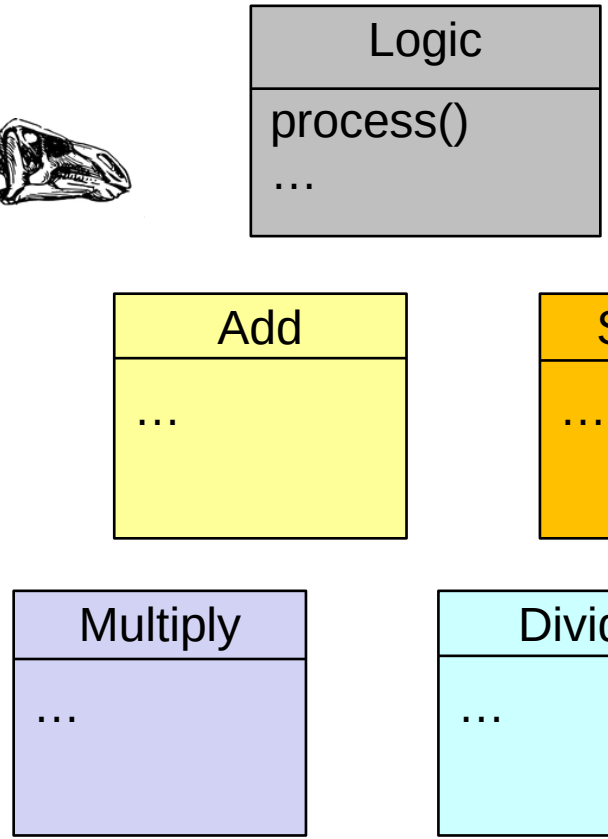
Steps for Getting Started

- Launch Android Studio & load the project



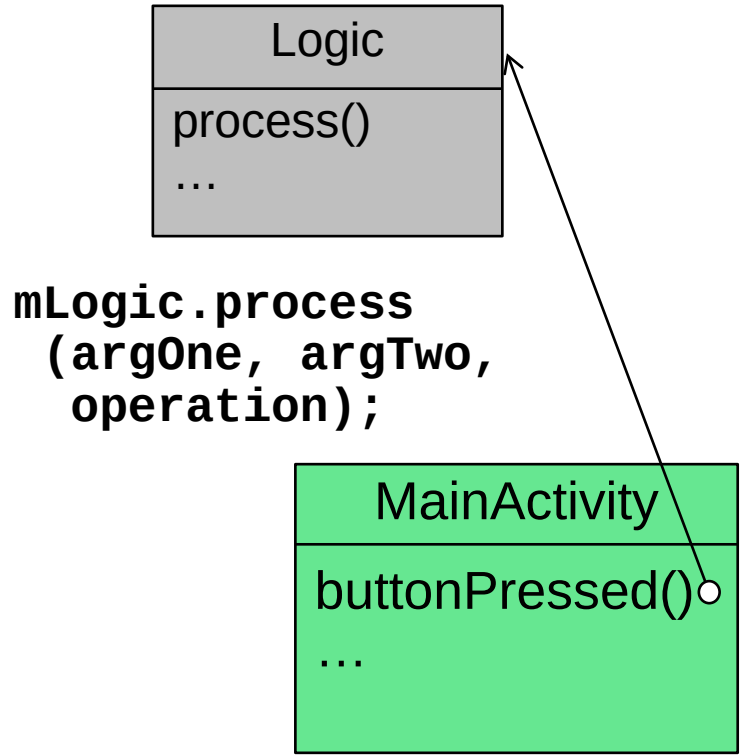
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes



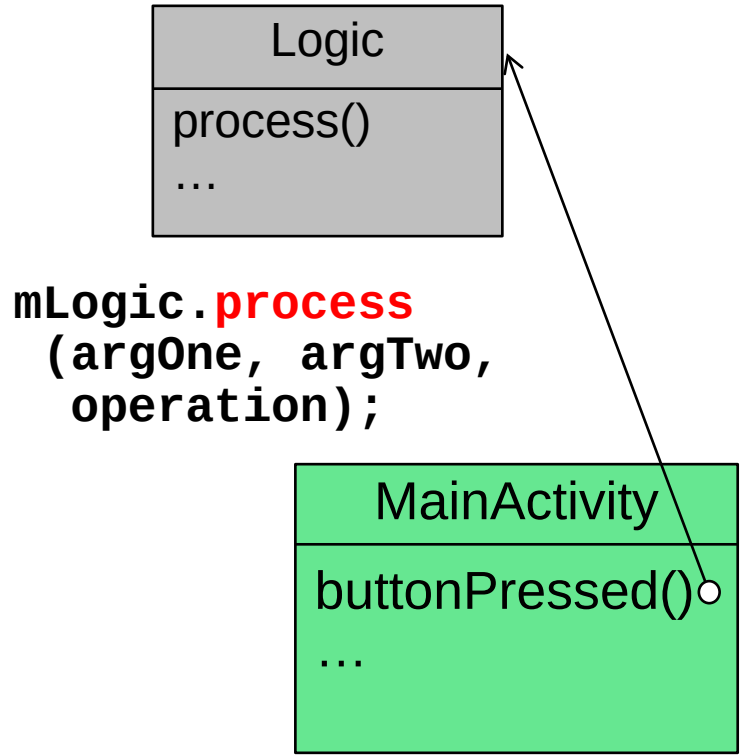
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI



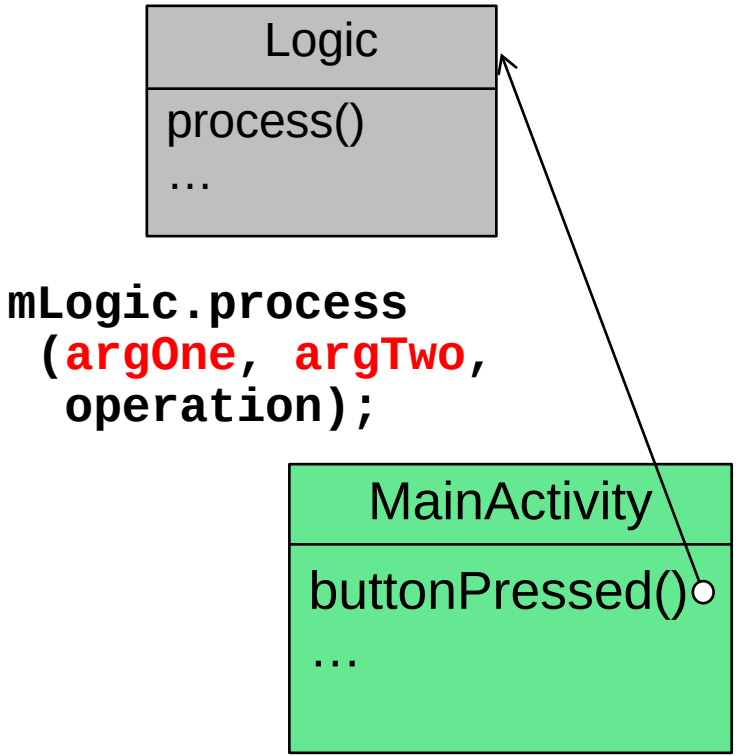
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI



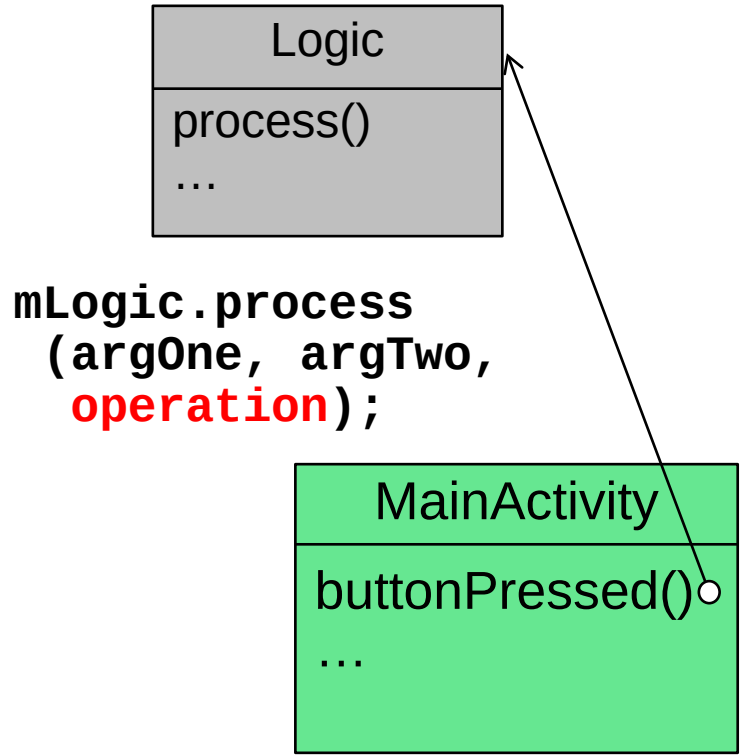
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
    - the two integers upon which to perform the computation



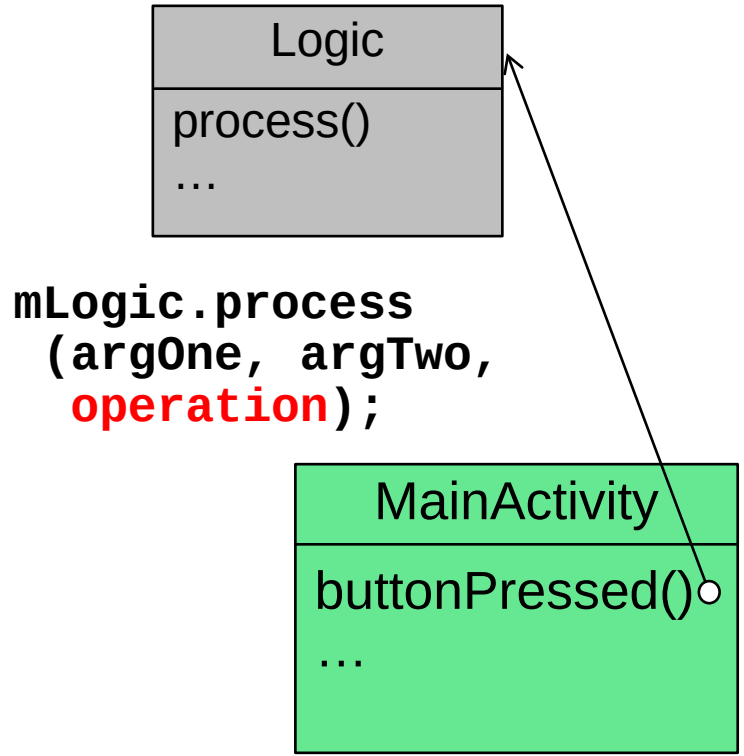
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
    - the two integers upon which to perform the computation
    - an integer value indicating the operation to perform



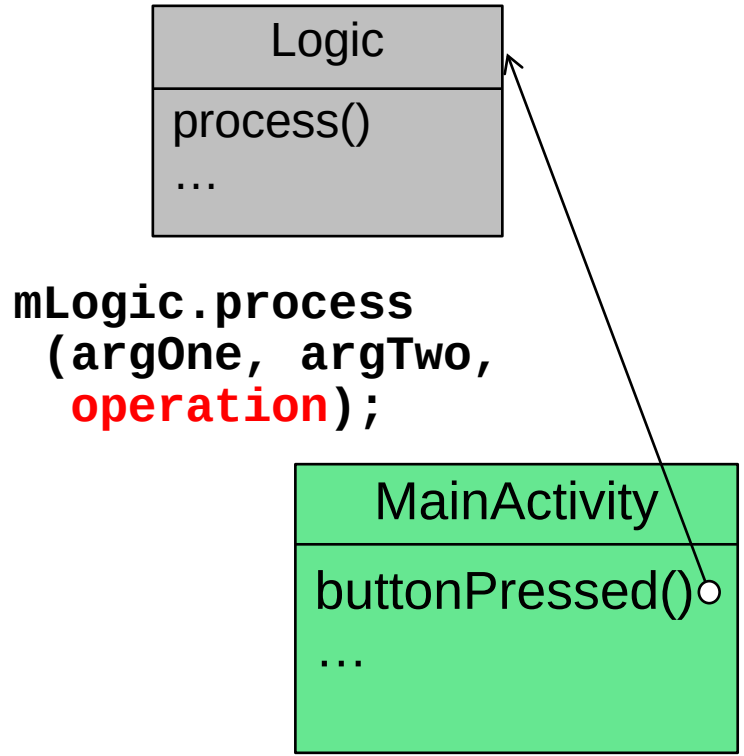
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
    - the two integers upon which to perform the computation
    - an integer value indicating the operation to perform
      - 1=addition



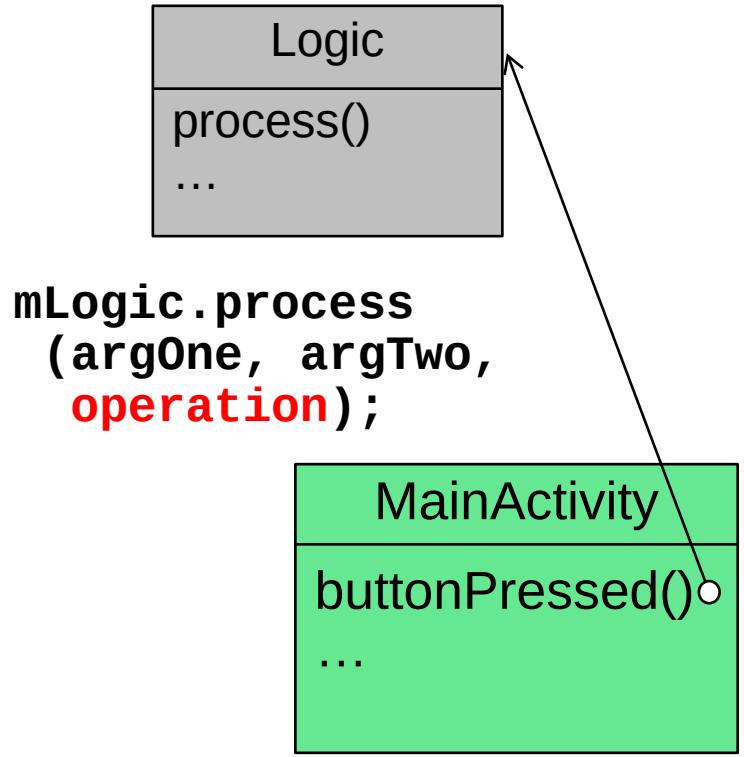
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
    - the two integers upon which to perform the computation
    - an integer value indicating the operation to perform
      - 1=addition
      - 2=subtraction



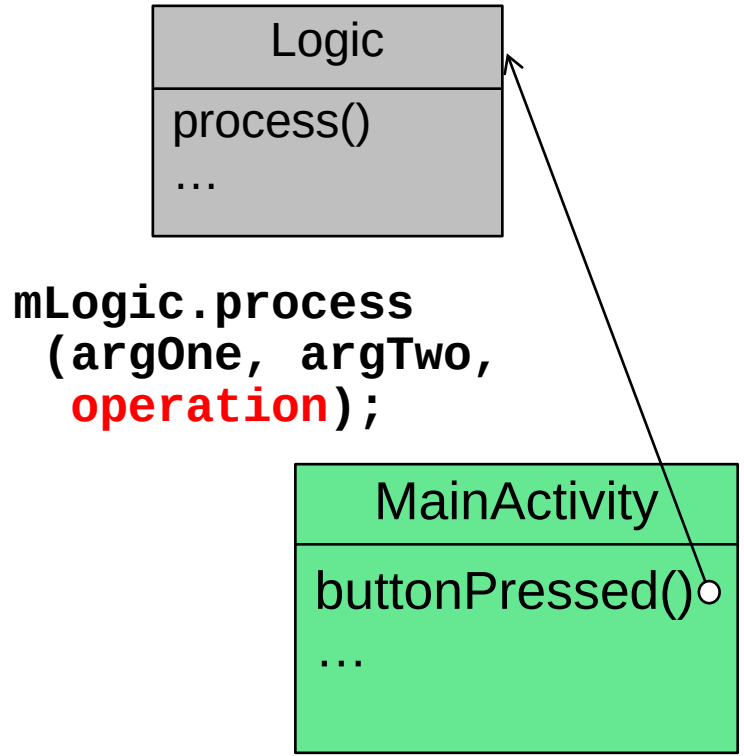
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
    - the two integers upon which to perform the computation
    - an integer value indicating the operation to perform
      - 1=addition
      - 2=subtraction
      - 3=multiplication



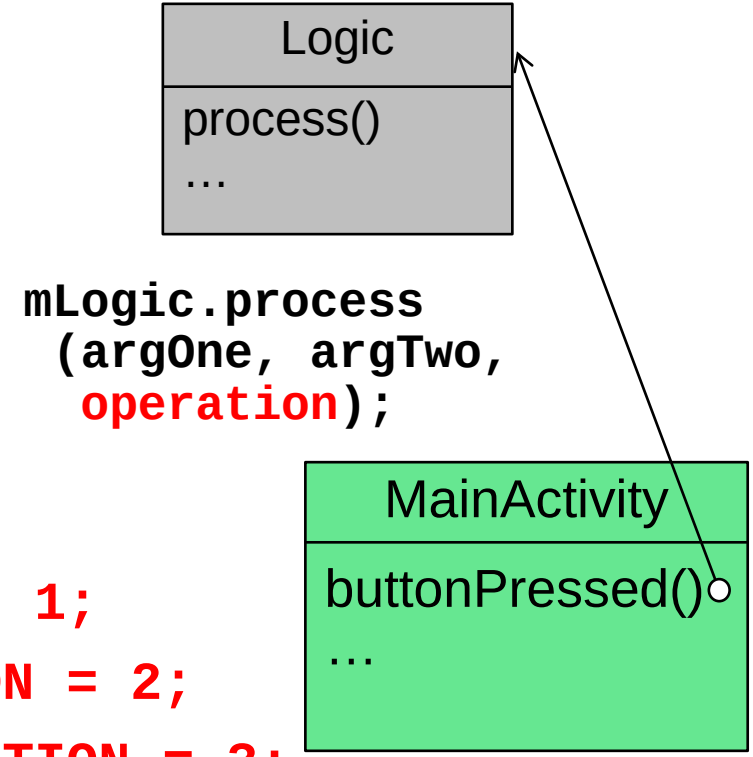
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
    - the two integers upon which to perform the computation
    - an integer value indicating the operation to perform
      - 1=addition
      - 2=subtraction
      - 3=multiplication
      - 4=division



Steps for Getting Started

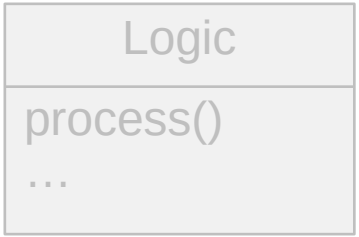
- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
    - the two integers upon which to perform the computation
    - an integer value indicating the operation to perform
      - `static final int ADDITION = 1;`
      - `static final int SUBTRACTION = 2;`
      - `static final int MULTIPLICATION = 3;`
      - `static final int DIVISION = 4;`

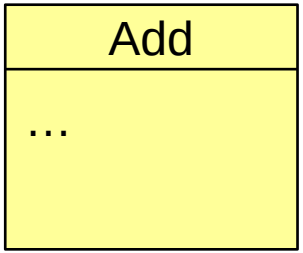


Use symbolic constants for these values, rather than “magic numbers”

Steps for Getting Started

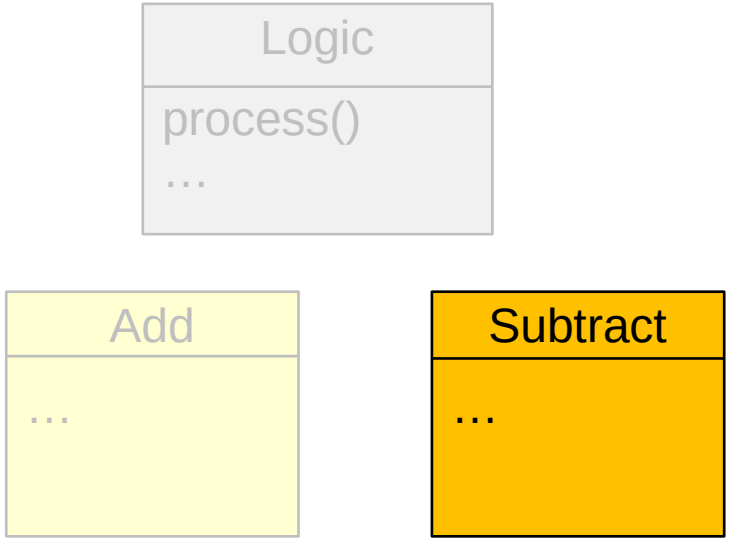
- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
  - The Add.java file contains an empty class named Add





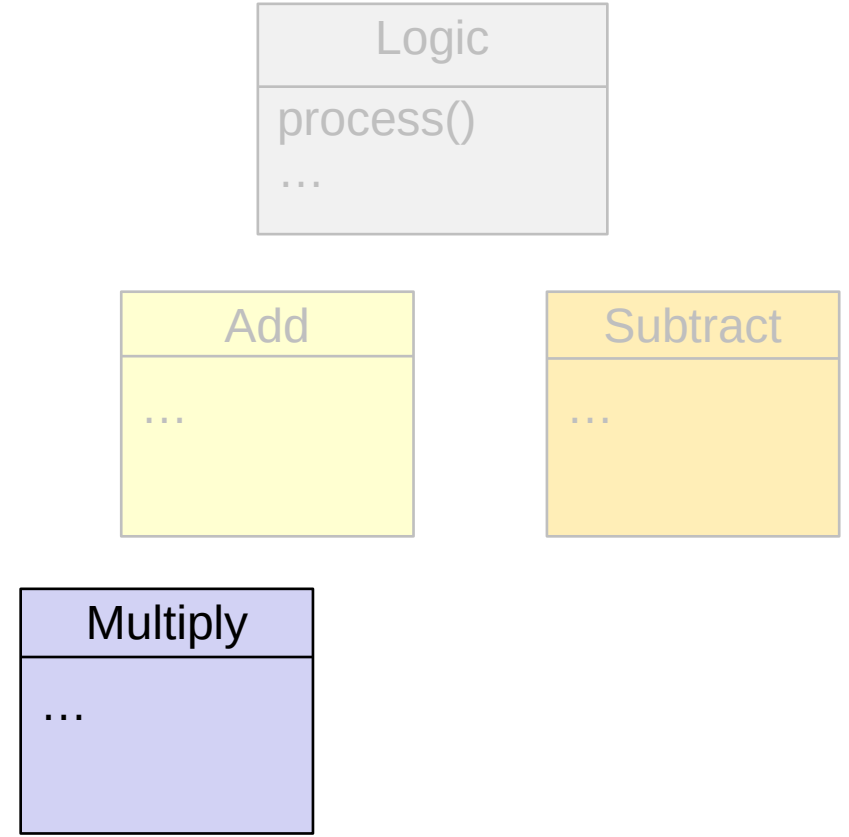
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
  - The Add.java file contains an empty class named Add
  - The Subtract.java file contains an empty class named Subtract



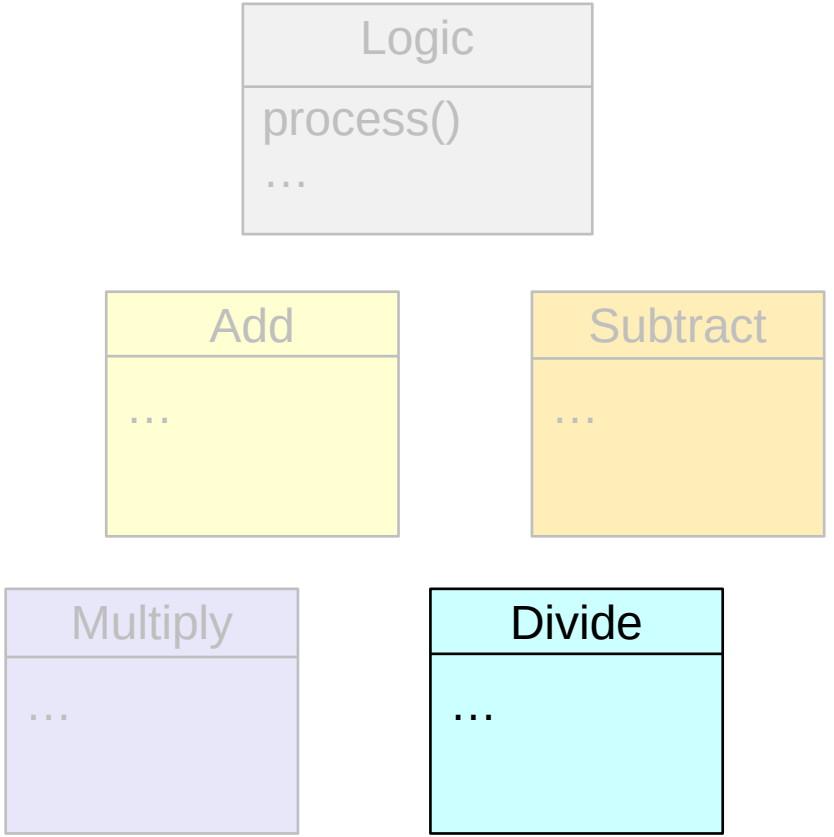
Steps for Getting Started

- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
  - The Add.java file contains an empty class named Add
  - The Subtract.java file contains an empty class named Subtract
  - The Multiply.java file contains an empty class named Multiply



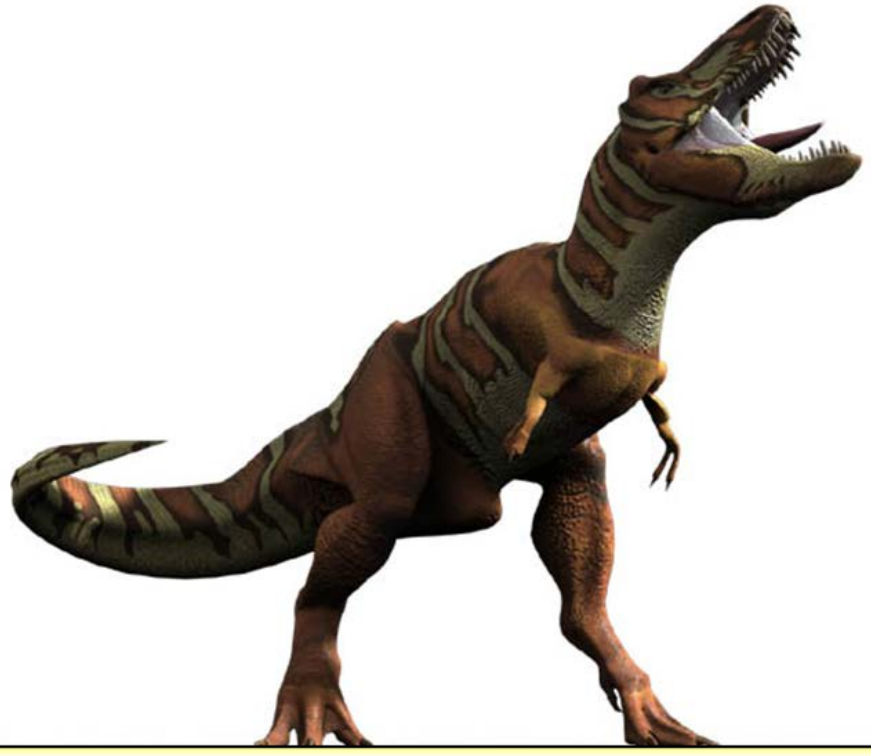
Steps for Getting Started

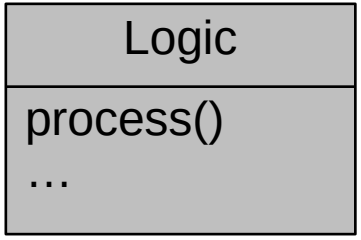
- This project supplies you with 5 skeleton files containing Java classes
  - Logic.java contains process(), which receives 3 entities passed from UI
  - The Add.java file contains an empty class named Add
  - The Subtract.java file contains an empty class named Subtract
  - The Multiply.java file contains an empty class named Multiply
  - The Divide.java file contains an empty class named Divide

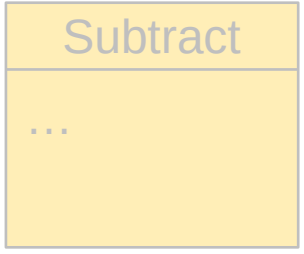


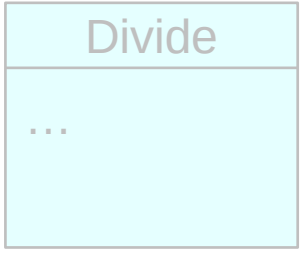
Steps for Getting Started

- You should start by modifying Logic.java









Add your implementation where comment says “TODO – start your code here”

Steps for Getting Started

- You should start by modifying Logic.java
  - Follow similar steps for other *.java files until you're done

Logic

process()

…

Add

…

Subtract

…

Multiply

…

Divide

…

Add your implementation where comment says "TODO – start your code here"

Steps for Getting Started

- You should start by modifying Logic.java
  - Follow similar steps for other *.java files until you're done

Logic
process() …

MainActivity
…

Add
…

Subtract
…

Multiply
…

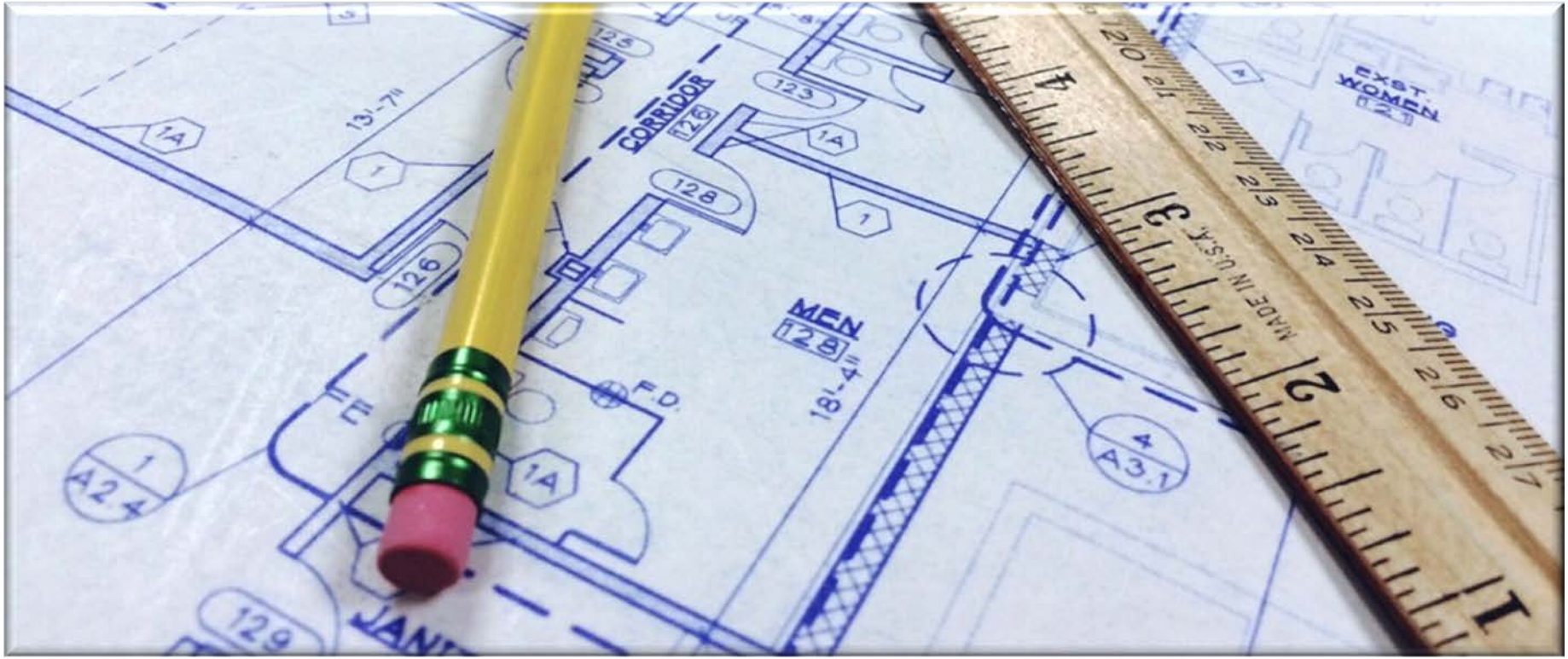
Divide
…

See upcoming lesson on "Mini-Project Assignment Walkthrough"

Guidelines for Structuring Your Solution

Guidelines for Structuring Your Solution

- There are two types of guidelines for structuring your solution



Guidelines for Structuring Your Solution

- *Source code design*

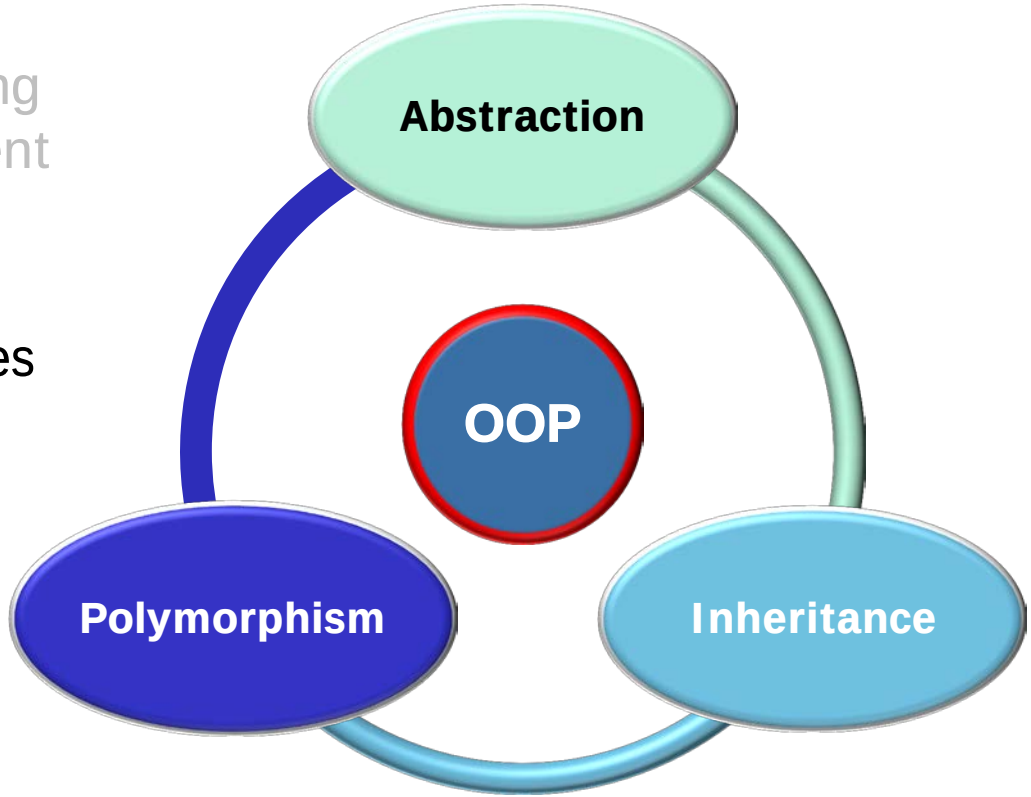
Guidelines for Structuring Your Solution

- *Source code design*
  - Don’t structure your code using a multi-branch if/else statement in the process() method

This design becomes unmanageable for future extensions of the calculator app

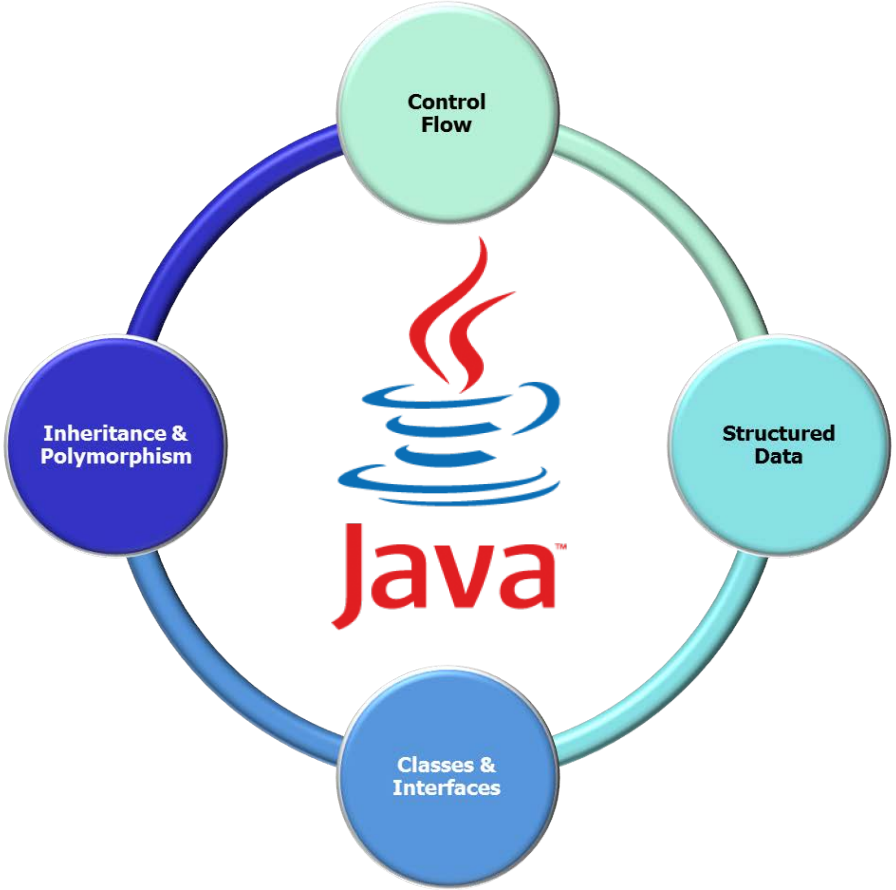
Guidelines for Structuring Your Solution

- *Source code design*
  - Don't structure your code using a multi-branch if/else statement in the process() method
  - Instead, create an *object-oriented* solution that simplifies extensibility & refactoring



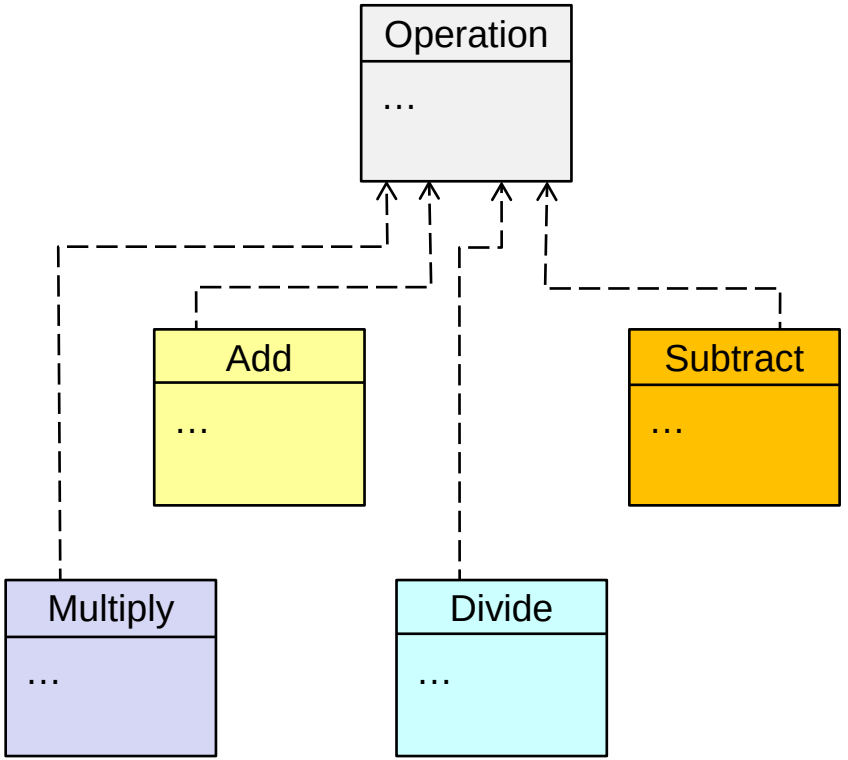
Guidelines for Structuring Your Solution

- *Source code design*
  - Don't structure your code using a multi-branch if/else statement in the process() method
  - Instead, create an *object-oriented* solution that simplifies extensibility & refactoring
  - To receive full credit, you must apply Java language features taught in recent modules



Guidelines for Structuring Your Solution

- *Source code design*
  - Don't structure your code using a multi-branch if/else statement in the process() method
  - Instead, create an *object-oriented* solution that simplifies extensibility & refactoring
  - To receive full credit, you must apply Java language features taught in recent modules
  - Consider defining a Java interface that these four classes implement



Guidelines for Structuring Your Solution

- *Source code aesthetics*

```
public void downloadAndDisplay(final DownloadConte
        // Create an AsyncTask to download an image i
        // and display it to the user in the UI Thread.
        mDownloader =  new AsyncTask<String, Void,
            /**
             * Called by the AsyncTask framework in the
             * perform initialization actions.
             */
            protected void onPreExecute() {
                // Show the toast before starting the dow
                // background Thread.
                downloadContext.showToast("downloadin
            }

            /**
             * Download a bitmap image in a backgroun
             */
            protected Bitmap doInBackground(String... u
                // Download the image, which can block.
                return downloadContext.downloadBitmap(
            }

            /**
             * Called after an operation executing in the
             * completes to set the bitmap image to an i
             * dismiss the progress dialog.
             */
            protected void onPostExecute(Bitmap image)
                // Display the downloaded image to the u
                downloadContext.displayBitmap(image);
            }
        }.execute(downloadContext.getUrl());
    }
```

Guidelines for Structuring Your Solution

- *Source code aesthetics*
  - Indent code consistently

```
public void downloadAndDisplay(final DownloadConte
        // Create an AsyncTask to download an image i
        // and display it to the user in the UI Thread.
        mDownloader =  new AsyncTask<String, Void,
            /**
             * Called by the AsyncTask framework in the
             * perform initialization actions.
             */
            protected void onPreExecute() {
                // Show the toast before starting the dow
                // background Thread.
                downloadContext.showToast("downloadin
            }

            /**
             * Download a bitmap image in a backgroun
             */
            protected Bitmap doInBackground(String... u
                // Download the image, which can block.
                return downloadContext.downloadBitmap(
            }

            /**
             * Called after an operation executing in the
             * completes to set the bitmap image to an i
             * dismiss the progress dialog.
             */
            protected void onPostExecute(Bitmap image)
                // Display the downloaded image to the u
                downloadContext.displayBitmap(image);
            }
        }.execute(downloadContext.getUrl());
    }
```

Guidelines for Structuring Your Solution

- *Source code aesthetics*
  - Indent code consistently
  - Keep lines of code <= 80 characters long

```
public void downloadAndDisplay(final DownloadConte
        // Create an AsyncTask to download an image i
        // and display it to the user in the UI Thread.
        mDownloader =  new AsyncTask<String, Void,
            /**
             * Called by the AsyncTask framework in the
             * perform initialization actions.
             */
            protected void onPreExecute() {
                // Show the toast before starting the dow
                // background Thread.
                downloadContext.showToast("downloadin
            }

            /**
             * Download a bitmap image in a backgroun
             */
            protected Bitmap doInBackground(String... u
                // Download the image, which can block.
                return downloadContext.downloadBitmap(
            }

            /**
             * Called after an operation executing in the
             * completes to set the bitmap image to an i
             * dismiss the progress dialog.
             */
            protected void onPostExecute(Bitmap image)
                // Display the downloaded image to the u
                downloadContext.displayBitmap(image);
            }
        }.execute(downloadContext.getUrl());
    }
```

Guidelines for Structuring Your Solution

- *Source code aesthetics*
  - Indent code consistently
  - Keep lines of code <= 80 characters long
  - Use an easy to read & maintain style

```
public void downloadAndDisplay(final DownloadConte
        // Create an AsyncTask to download an image i
        // and display it to the user in the UI Thread.
        mDownloader =  new AsyncTask<String, Void,
            /**
             * Called by the AsyncTask framework in the
             * perform initialization actions.
             */
            protected void onPreExecute() {
                // Show the toast before starting the dow
                // background Thread.
                downloadContext.showToast("downloadin
            }

            /**
             * Download a bitmap image in a backgroun
             */
            protected Bitmap doInBackground(String... u
                // Download the image, which can block.
                return downloadContext.downloadBitmap(
            }

            /**
             * Called after an operation executing in the
             * completes to set the bitmap image to an i
             * dismiss the progress dialog.
             */
            protected void onPostExecute(Bitmap image)
                // Display the downloaded image to the u
                downloadContext.displayBitmap(image);
            }
        }.execute(downloadContext.getUrl());
    }
```

Guidelines for Structuring Your Solution

- *Source code aesthetics*
  - Indent code consistently
  - Keep lines of code <= 80 characters long
  - Use an easy to read & maintain style
    - Create helper methods

```
public void downloadAndDisplay(final DownloadConte
        // Create an AsyncTask to download an image i
        // and display it to the user in the UI Thread.
        mDownloader =  new AsyncTask<String, Void,
            /**
             * Called by the AsyncTask framework in the
             * perform initialization actions.
             */
            protected void onPreExecute() {
                // Show the toast before starting the dow
                // background Thread.
                downloadContext.showToast("downloadin
            }

            /**
             * Download a bitmap image in a backgroun
             */
            protected Bitmap doInBackground(String... u
                // Download the image, which can block.
                return downloadContext.downloadBitmap(
            }

            /**
             * Called after an operation executing in the
             * completes to set the bitmap image to an i
             * dismiss the progress dialog.
             */
            protected void onPostExecute(Bitmap image)
                // Display the downloaded image to the u
                downloadContext.displayBitmap(image);
            }
        }.execute(downloadContext.getUrl());
    }
```

Guidelines for Structuring Your Solution

- *Source code aesthetics*
  - Indent code consistently
  - Keep lines of code <= 80 characters long
  - Use an easy to read & maintain style
    - Create helper methods
    - Using meaningful variable/method names

```
public void downloadAndDisplay(final DownloadConte
      // Create an AsyncTask to download an image i
      // and display it to the user in the UI Thread.
      mDownloader =  new AsyncTask<String, Void,
         /**
          * Called by the AsyncTask framework in the
          * perform initialization actions.
          */
         protected void onPreExecute() {
            // Show the toast before starting the dow
            // background Thread.
            downloadContext.showToast("downloadin
         }

         /**
          * Download a bitmap image in a backgroun
          */
         protected Bitmap doInBackground(String... u
            // Download the image, which can block.
            return downloadContext.downloadBitmap(
         }

         /**
          * Called after an operation executing in the
          * completes to set the bitmap image to an i
          * dismiss the progress dialog.
          */
         protected void onPostExecute(Bitmap image)
            // Display the downloaded image to the u
            downloadContext.displayBitmap(image);
         }
      }.execute(downloadContext.getUrl());
   }
```

Guidelines for Structuring Your Solution

- *Source code aesthetics*
  - Indent code consistently
  - Keep lines of code <= 80 characters long
  - Use an easy to read & maintain style
    - Create helper methods
    - Using meaningful variable/method names
    - Make the code readable

```
public void downloadAndDisplay(final DownloadConte
      // Create an AsyncTask to download an image i
      // and display it to the user in the UI Thread.
      mDownloader =  new AsyncTask<String, Void,
         /**
          * Called by the AsyncTask framework in the
          * perform initialization actions.
          */
         protected void onPreExecute() {
            // Show the toast before starting the dow
            // background Thread.
            downloadContext.showToast("downloadin
         }

         /**
          * Download a bitmap image in a backgroun
          */
         protected Bitmap doInBackground(String... u
            // Download the image, which can block.
            return downloadContext.downloadBitmap(
         }

         /**
          * Called after an operation executing in the
          * completes to set the bitmap image to an i
          * dismiss the progress dialog.
          */
         protected void onPostExecute(Bitmap image)
            // Display the downloaded image to the u
            downloadContext.displayBitmap(image);
         }
      }.execute(downloadContext.getUrl());
   }
```

Guidelines for Structuring Your Solution

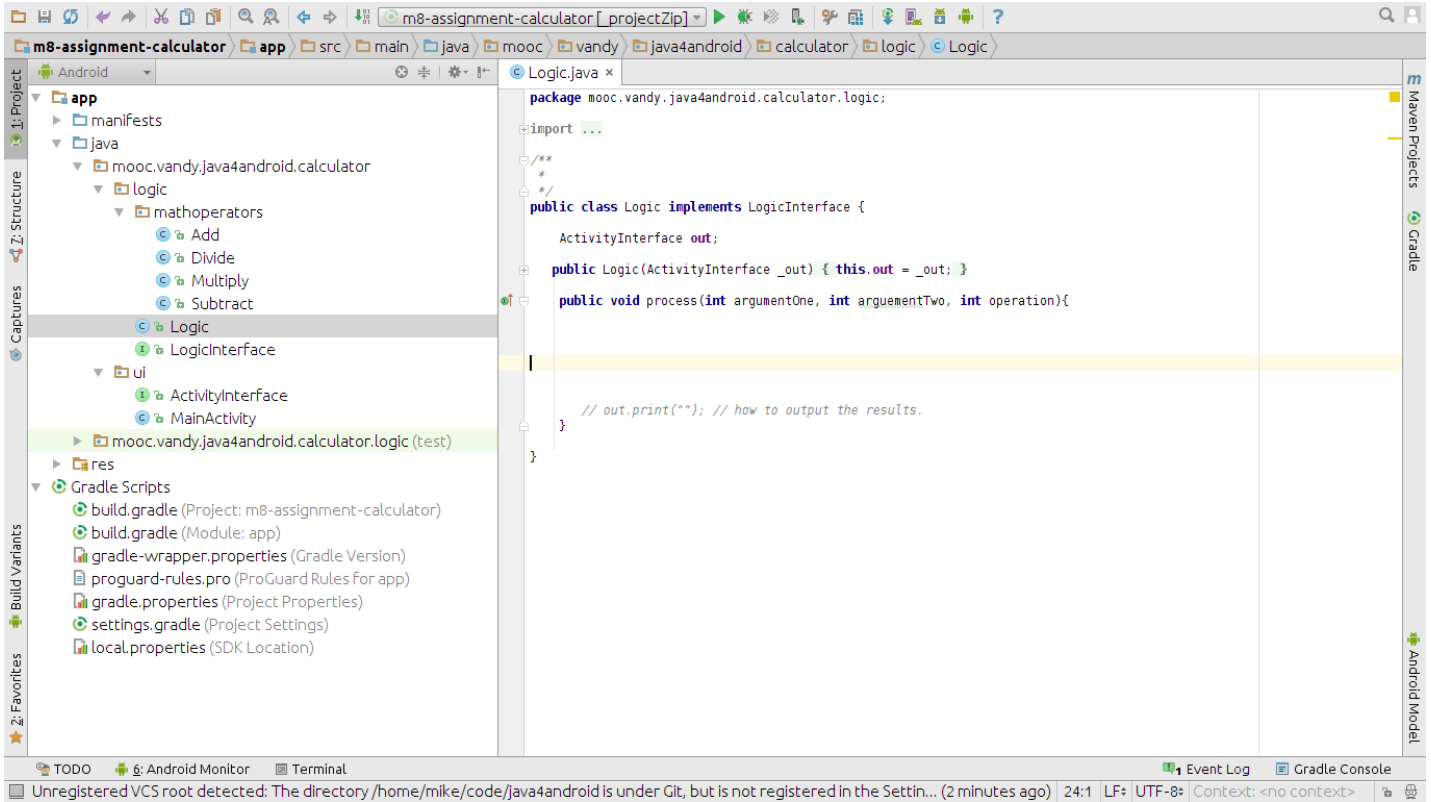
- *Source code aesthetics*
  - Indent code consistently
  - Keep lines of code <= 80 characters long
  - Use an easy to read & maintain style
    - Create helper methods
    - Using meaningful variable/method names
    - Make the code readable
    - Explain code with useful comments

```
public void downloadAndDisplay(final DownloadConte
        // Create an AsyncTask to download an image i
        // and display it to the user in the UI Thread.
        mDownloader =  new AsyncTask<String, Void,
            /**
             * Called by the AsyncTask framework in the
             * perform initialization actions.
             */
            protected void onPreExecute() {
                // Show the toast before starting the dow
                // background Thread.
                downloadContext.showToast("downloadin
            }

            /**
             * Download a bitmap image in a backgroun
             */
            protected Bitmap doInBackground(String... u
                // Download the image, which can block.
                return downloadContext.downloadBitmap(
            }

            /**
             * Called after an operation executing in the
             * completes to set the bitmap image to an i
             * dismiss the progress dialog.
             */
            protected void onPostExecute(Bitmap image)
                // Display the downloaded image to the u
                downloadContext.displayBitmap(image);
            }
        }.execute(downloadContext.getUrl());
    }
```

Steps for Submitting Your Mini-Project Solution

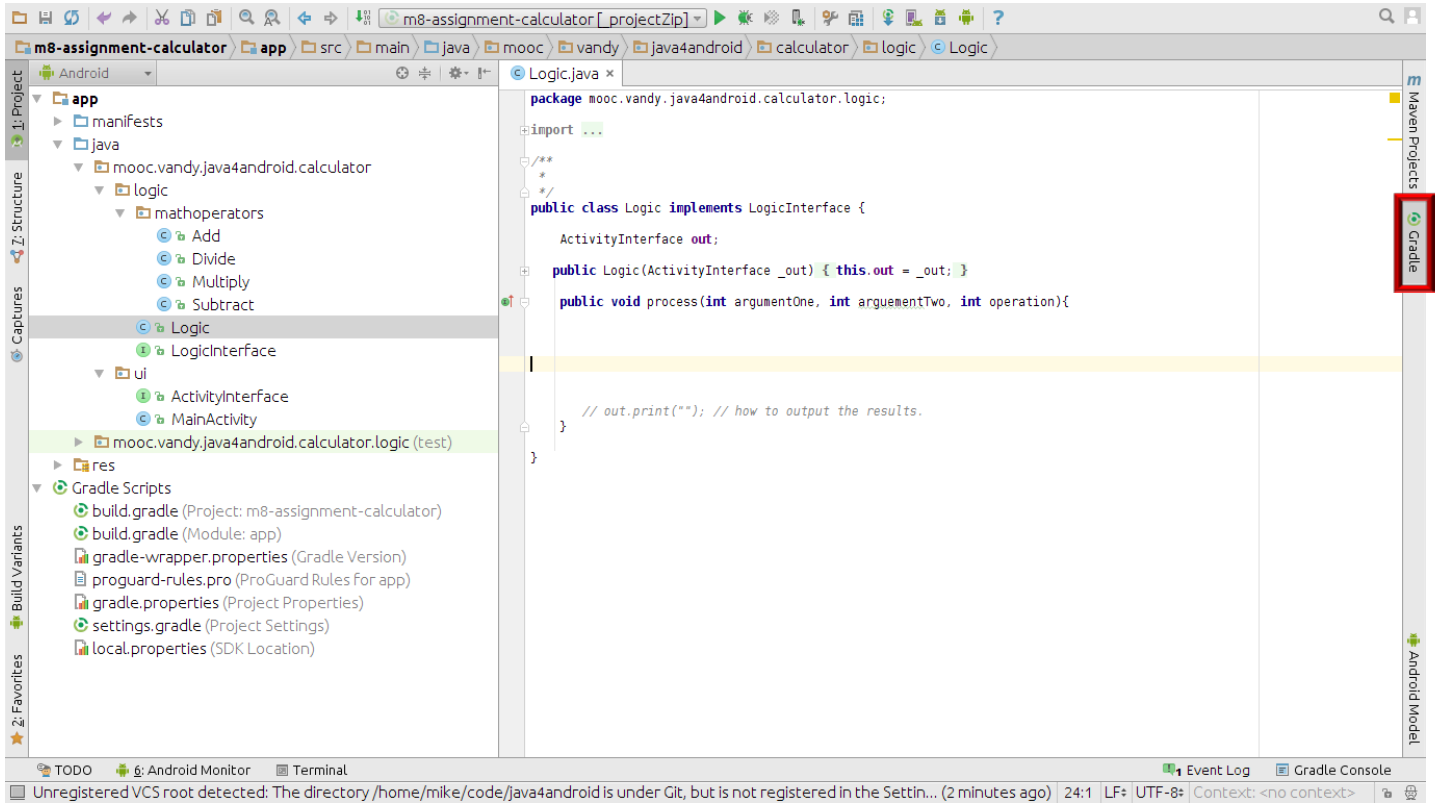
Steps for Submitting Your Mini-Project Solution

- Submit a zip file with all the necessary Java & Android Studio project files



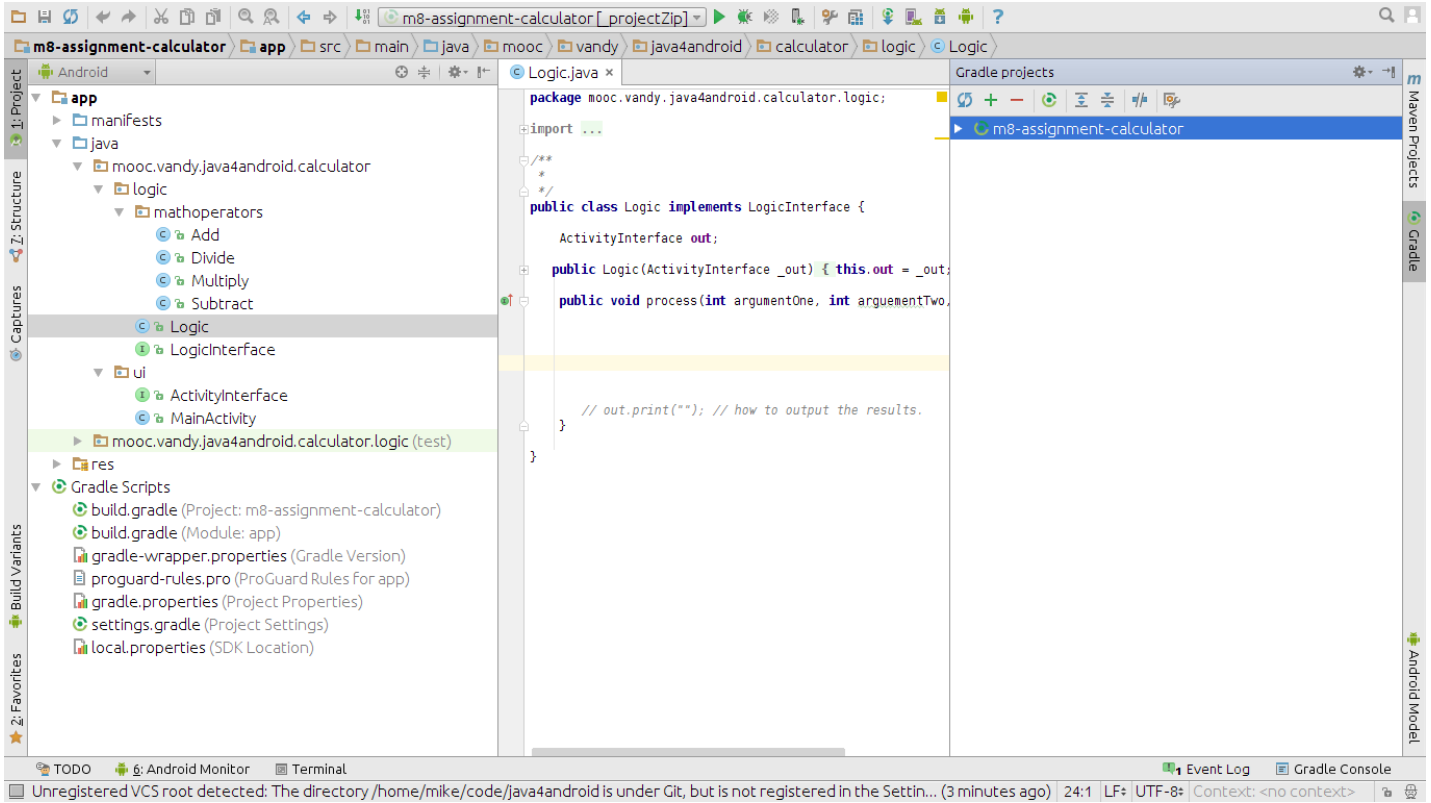
Steps for Submitting Your Mini-Project Solution

- Submit a zip file with all the necessary Java & Android Studio project files



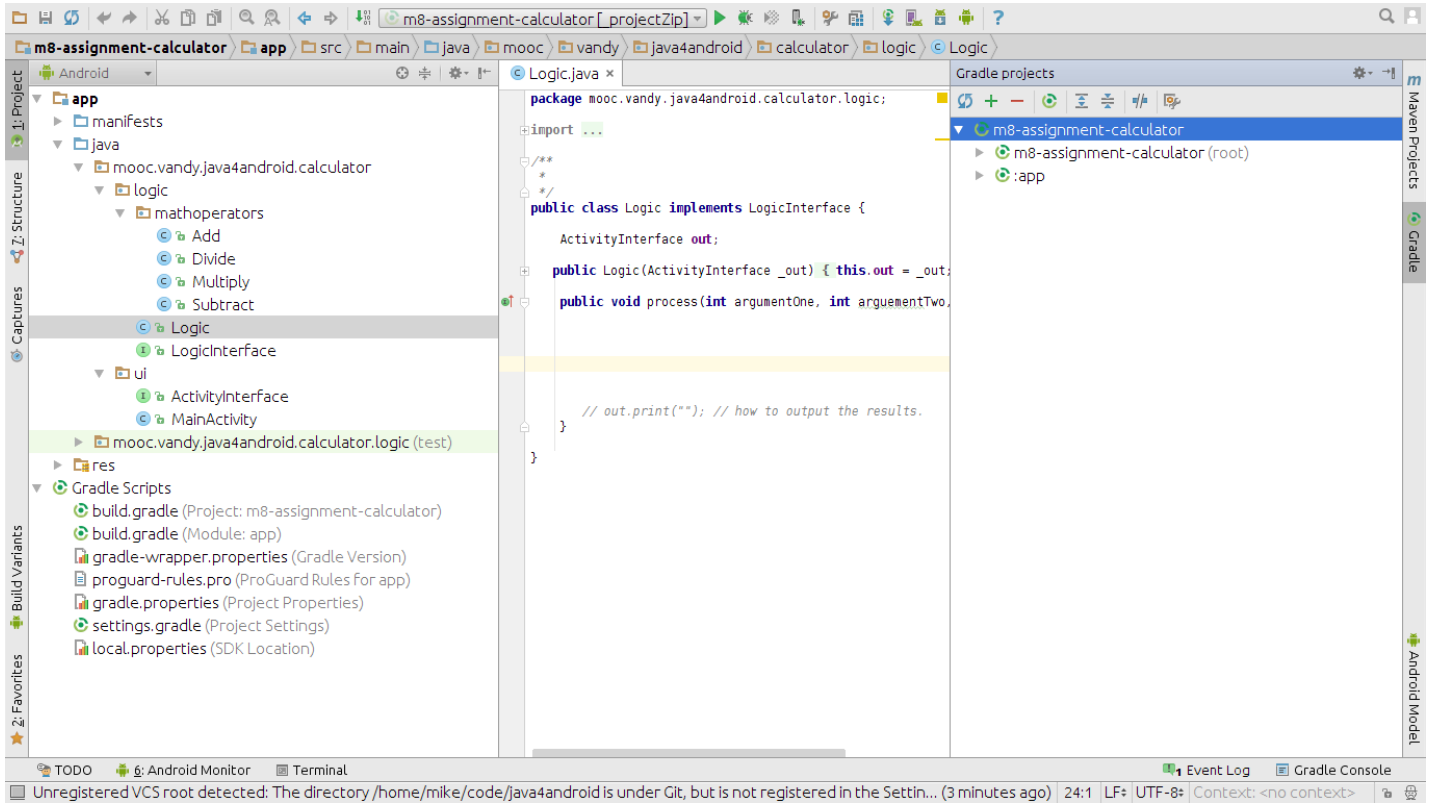
Steps for Submitting Your Mini-Project Solution

- Submit a zip file with all the necessary Java & Android Studio project files



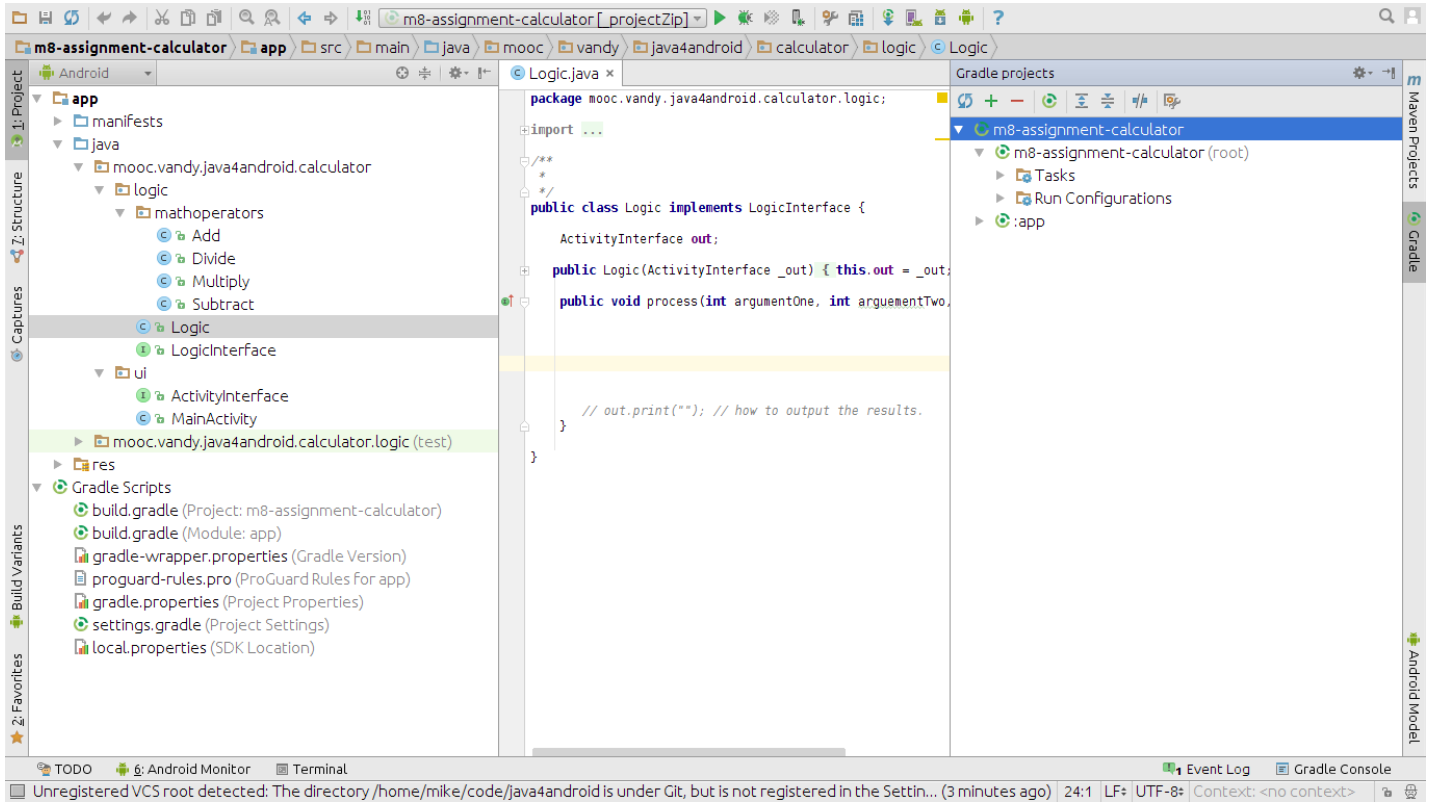
Steps for Submitting Your Mini-Project Solution

- Submit a zip file with all the necessary Java & Android Studio project files



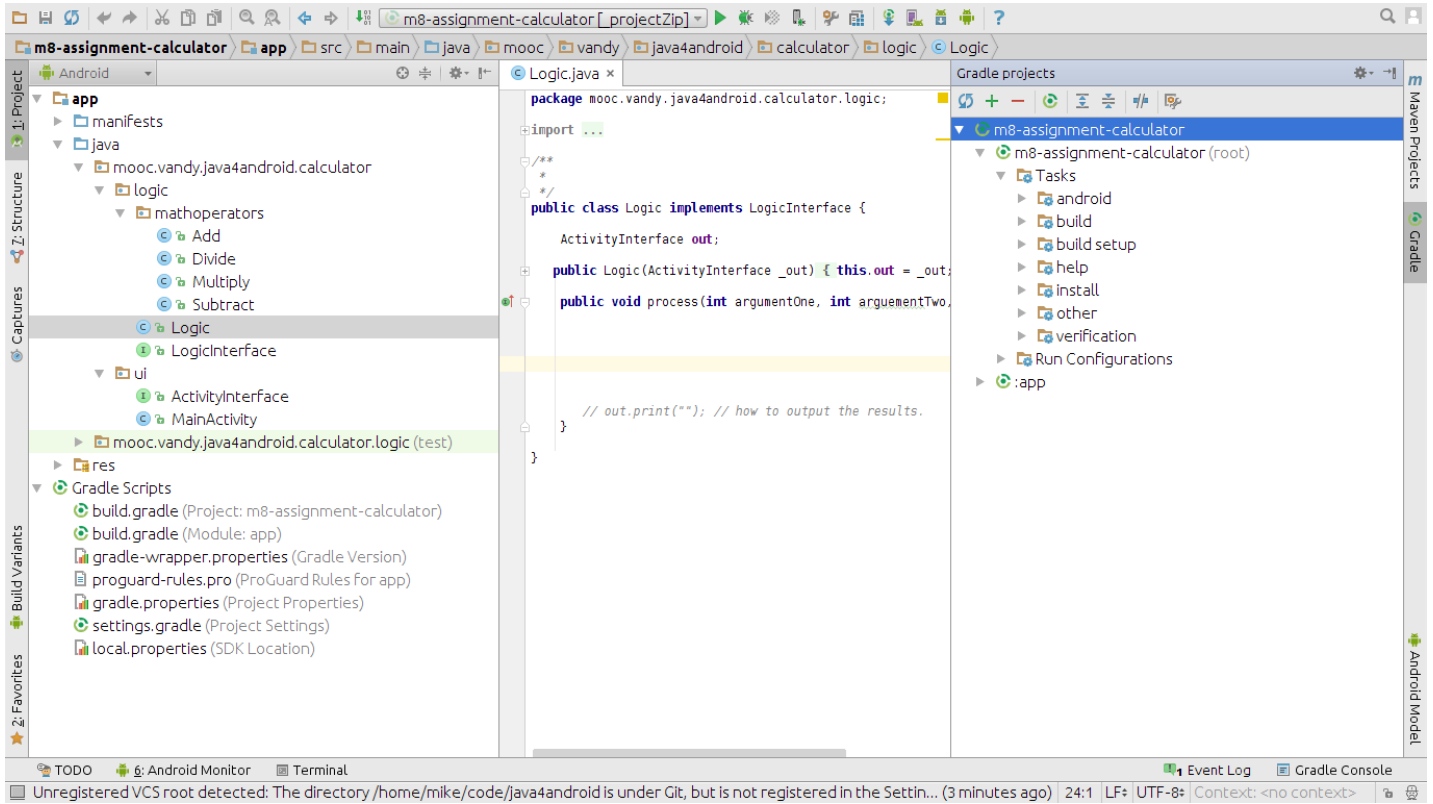
Steps for Submitting Your Mini-Project Solution

- Submit a zip file with all the necessary Java & Android Studio project files



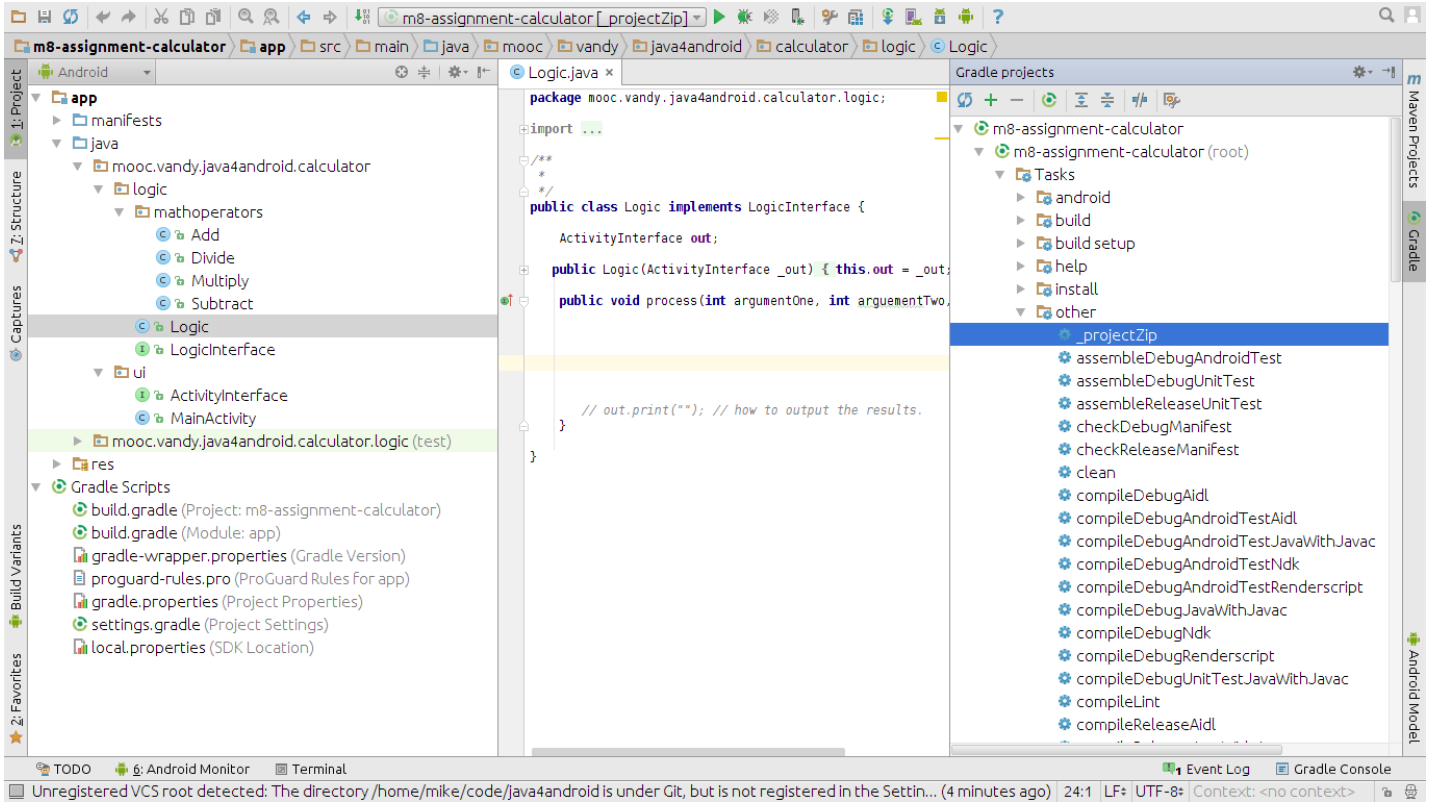
Steps for Submitting Your Mini-Project Solution

- Submit a zip file with all the necessary Java & Android Studio project files



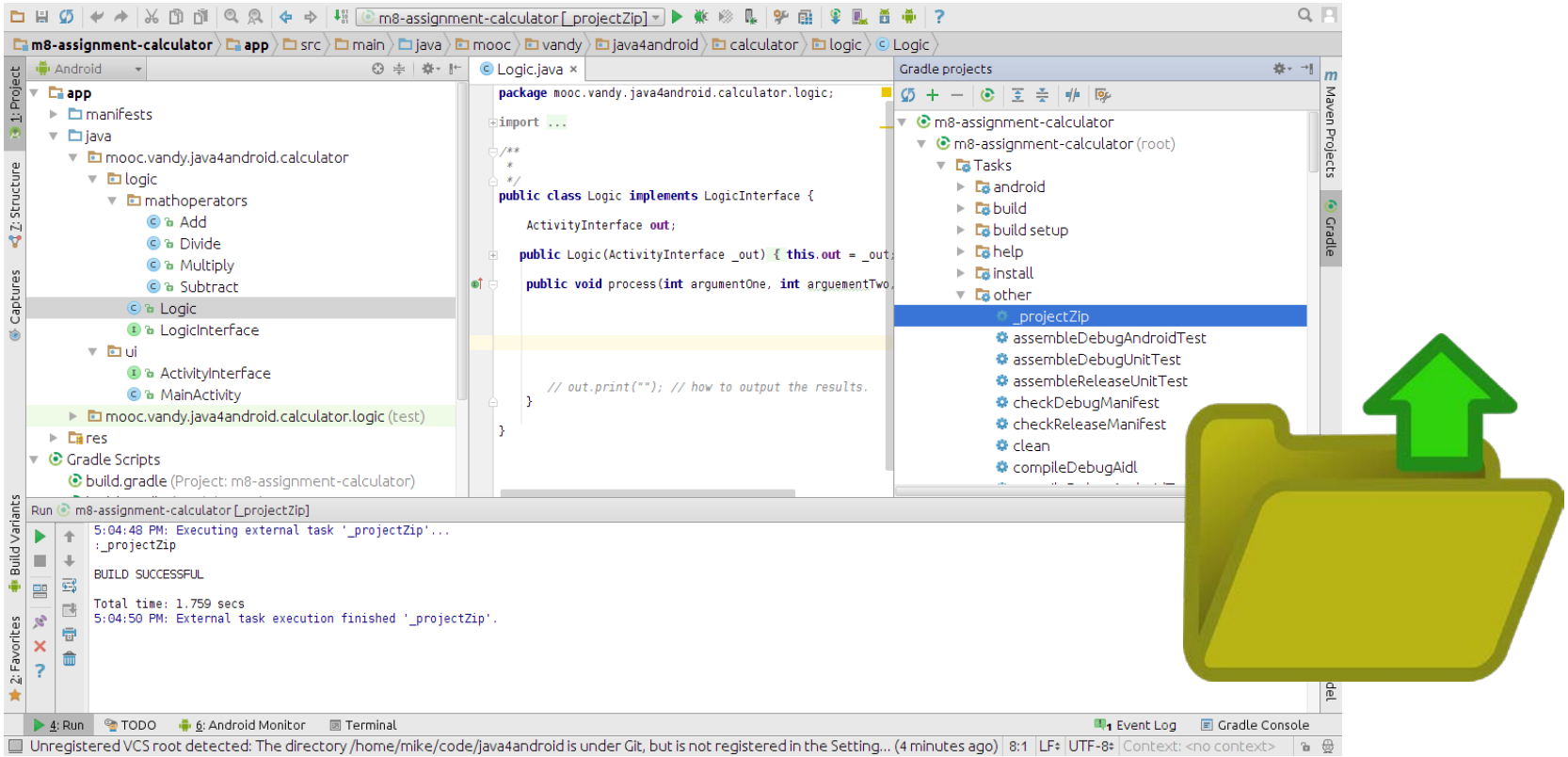
Steps for Submitting Your Mini-Project Solution

- Submit a zip file with all the necessary Java & Android Studio project files



Steps for Submitting Your Mini-Project Solution

- Submit a zip file with all the necessary Java & Android Studio project files



Steps for Assessing Peer Solutions

Steps for Assessing Peer Solutions

- This mini-project will be purely peer assessed

See learner.coursera.help/hc/en-us/articles/201893769-Peer-Review-Assignments

Steps for Assessing Peer Solutions

- There are two steps involved

Steps for Assessing Peer Solutions

- There are two steps involved
  1. Submit your assignment as discussed earlier

Steps for Assessing Peer Solutions

- There are two steps involved
  1. Submit your assignment as discussed earlier
     - At which point you'll get a video of our solution

See conditional release video on "Mini-Project Solution Walkthrough"

Steps for Assessing Peer Solutions

- There are two steps involved
  1. Submit your assignment as discussed earlier
  2. Review 5 peer submissions

Steps for Assessing Peer Solutions

- There are two steps involved
  1. Submit your assignment as discussed earlier
  2. Review 5 peer submissions
     - Using the grading rubric that we supply

Steps for Assessing Peer Solutions

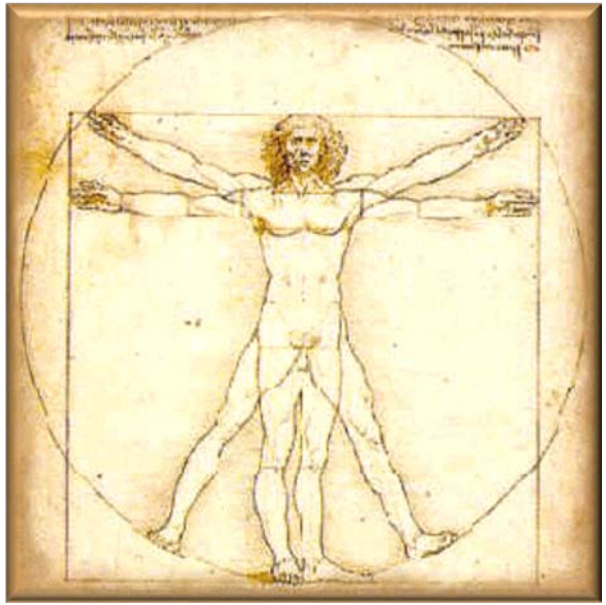
- Your final grade on the mini-project uses median scores you receive from peers

Steps for Assessing Peer Solutions

- Your final grade on the mini-project uses median scores you receive from peers
  - There's a 20% penalty for not evaluating peers

Steps for Assessing Peer Solutions

- Keep an open mind & focus on the positive in your peer evaluations



Steps for Assessing Peer Solutions

- Keep an open mind & focus on the positive in your peer evaluations
  - The goal is *not* to find every way to deduct points

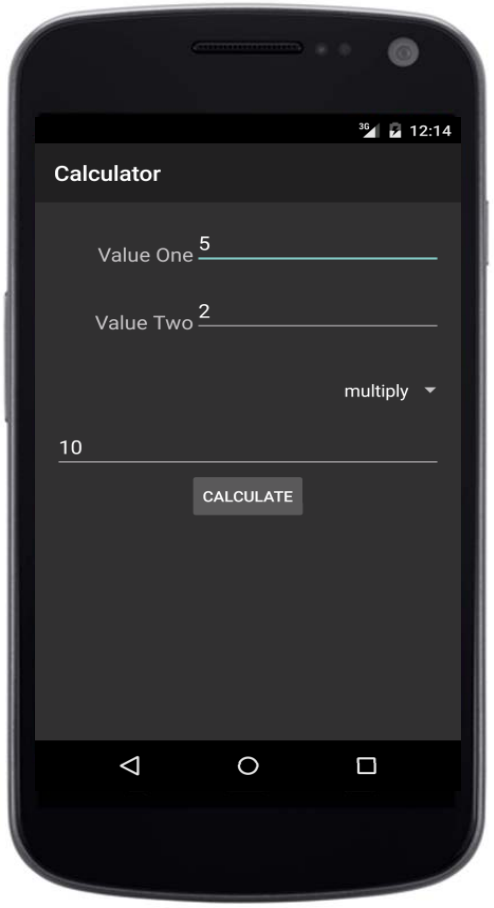
Steps for Assessing Peer Solutions

- Keep an open mind & focus on the positive in your peer evaluations
  - The goal is *not* to find every way to deduct points
  - Look for ways to give points when it's clear the submitter has given a good faith effort

Steps for Assessing Peer Solutions

- Keep an open mind & focus on the positive in your peer evaluations
  - The goal is *not* to find every way to deduct points
  - Look for ways to give points when it's clear the submitter has given a good faith effort
  - Error on the side of giving too many points, rather than giving too few

Java for Android: Calculator App



Java for Android: Calculator App

Learning objectives:

- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

Java for Android: Calculator App

Learning objectives:

- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

Java for Android: Calculator App

Learning objectives:

- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

Java for Android: Calculator App

Learning objectives:

- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

Java for Android: Calculator App

Learning objectives:

- Understand the requirements of the calculator app
- Know how to download the Android Studio project & files containing the app skeleton
- Be familiar with guidelines for structuring your solution
- Recognize how to submit your solution & assess solutions by other learners in the MOOC

Java for Android: Calculator App

Doug Schmidt

Julie Johnson

Mike Walker

Jerry Roth

Java for Android: Calculator App

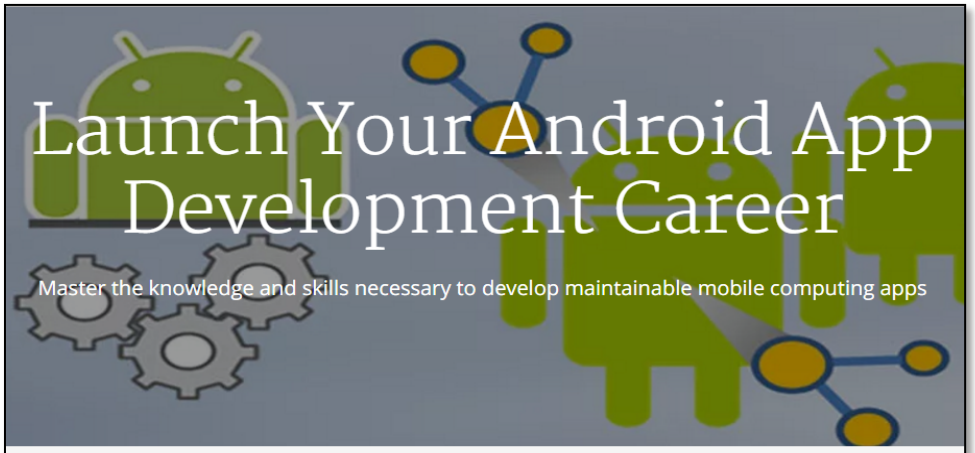
Doug Schmidt

Julie Johnson

Mike Walker

Jerry Roth

Java for Android: Calculator App



About This Specialization

This Specialization enables learners to successfully apply core Java programming languages features & software patterns needed to develop maintainable mobile apps comprised of core Android components, as well as fundamental Java I/O & persistence mechanisms. Learners who successfully complete this Specialization will be well-prepared to master the more advanced material in the subsequent "Mobile Cloud Computing with Android" Specialization.

The Capstone project will integrate the material from throughout the Specialization to exercise and assess the ability of learners to create an interesting Android app by applying knowledge and skills learned in previous MOOCs, including Java programming features, Android Studio tools, Android Activity components, Material Design, file I/O and data persistence, unit testing, and software patterns. The project itself will be similar in design goals to previous assignments, however it will provide less of the skeleton code than earlier MOOCs provide to enable more creativity to learners and greater opportunity for learners to customize the app.

Created by: VANDERBILT UNIVERSITY