Tutorial 2.

Card 24 Game (Part I)

2022-2023

COMP3330 Interactive Mobile Application Design and Programming

Dr. T.W. Chim (E-mail: twchim@cs.hku.hk)

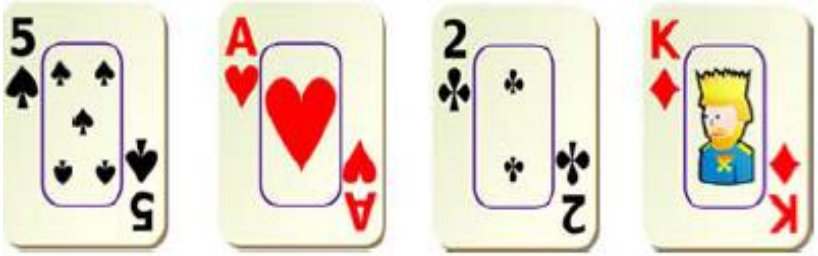
Department of Computer Science, The University of Hong Kong

Card 24 Game

- In this and the next tutorial, you will learn how to build an Android-based Card 24 game.
- This tutorial will focus on user interface design while the next will focus on programming logic.

Card 24 Game

- Card24 is an arithmetical card game in which the objective is to find a way to manipulate the numbers of four poker cards so that the end result is 24.
- Addition, subtraction, multiplication, or division, and sometimes other operations, may be used to make four numbers (from 1 to 13, where Ace, Jack, Queen, King represents 1, 11, 12, 13, respectively) equal 24.
- For example, for the four cards below, one of the possible solutions can be (13 – 5) * (1 + 2) = 24.



Application Specification

Start Game:

The application should randomly pick 4 non-duplicate cards from totally 52 poker cards (excluding two jokers cards) when the game starts. The selected cards are then displayed on the screen. Player can then choose these cards and arithmetic symbols to form the formula.

Application Specification

Formula:

Player is allowed to use 4 arithmetic symbols in the formula, including add “+”, subtract “-”, multiply “x”, and divide “/”. Player can also use the open bracket “(” and close bracket “)” to specify the order of calculation. NOTE that the formula should be evaluated in the order of “bracket items” before “multiplication/division” before “addition/subtraction”. For example, the formula can be “(1) + (11 – 2) x 6”, and the order of evaluating the formula should be:

(1) = 1

(11 – 2) = 9

9 x 6 = 54

1 + 54 = 55

Ans. = 55

Application Specification

Furthermore, some basic limitations can be included to prevent non-sense player input, like “()1+/12”. Here are some suggestions for these limitations:

- Cards or open bracket “(” can only be selected when it is the first item in the formula, or it follows an arithmetic symbol or an open bracket “(”.
- Arithmetic symbols “+”, “-”, “x”, “/” and the close bracket “)” can only be selected when the last previous selected item is card or a close bracket “)”.

Application Specification

Winning the game:

- The player wins when:
  - There is no error in the formula.
  - All four cards are selected exactly once.
  - The result of the formula equals 24.

Application Specification

Implementation details:

The layout should contain the following components:

- 4 Buttons for 4 random poker cards.
- 6 Buttons for 6 arithmetic symbols (“+”, “-”, “x”, “/”, “(”, “)”).
- 1 TextView for displaying the current formula.
- 2 Buttons for i) starting a new game and ii) clearing the formula.
- 1 equal “=” Button for evaluating the formula. After pressed, the player would be informed whether the answer is correct (player wins) or wrong (player loses).

Application Specification

During the implementation, you should take care of the following cases:

- Each of the 4 poker cards should only be selected once. Once pressed, the card button should take no more effect (disabled) when pressing the card again. Some indicators can be used to show whether the card is selected or not.
- When “new game” or “clear” buttons are pressed, some of the disabled button may be enabled again, and the current formula should be displayed accordingly.
- When the formula is not valid when pressing the “=” Button, an error message should be shown to tell the player that the answer is considered to be wrong (player loses).

Task 1

- Create a new project in Android Studio.
- Choose your own application name (e.g. “Card24”) and company domain (e.g. “hkucs”).
- Choose “Kotlin” as the programming language.
- Leave all other settings (including key file names) as default for simplicity.

Task 2

- Let's first work on the layout file ("activity_main.xml").
- Let's use nested layouts as on the right diagram to make the appearance better.
- The outer-most layout is a scroll view while all inner layouts are linear layouts.

ScrollView

- Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display.
- A ScrollView is a FrameLayout, meaning you should place one child in it containing the entire contents to scroll. This child may itself be a layout manager with a complex hierarchy of objects.

Example:

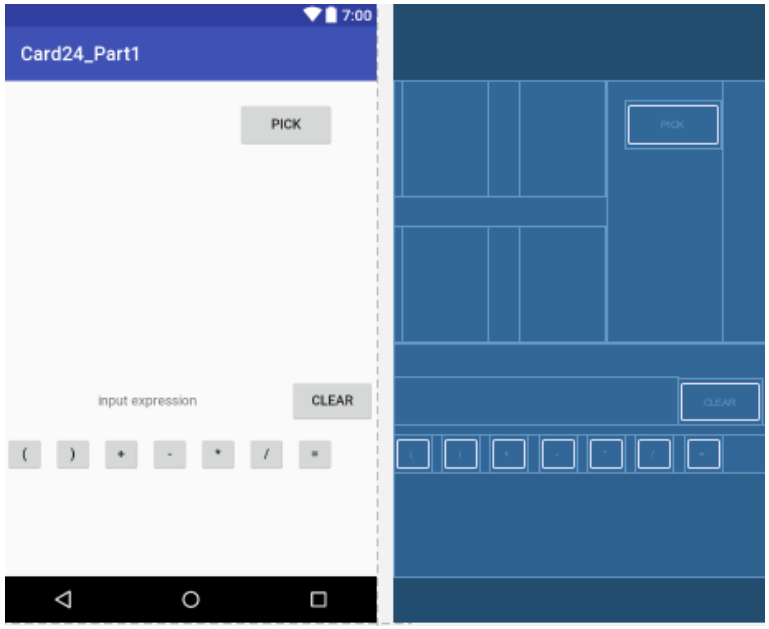
ScrollView

- Example:

```
<ScrollView
xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="fill_parent“
android:layout_height="fill_parent“
android:fillViewport="true">
        <RelativeLayout
        android:layout_width="fill_parent“
        android:layout_height="wrap_content" android:background="#ffffff">
        …
        </RelativeLayout>
</ScrollView>
```

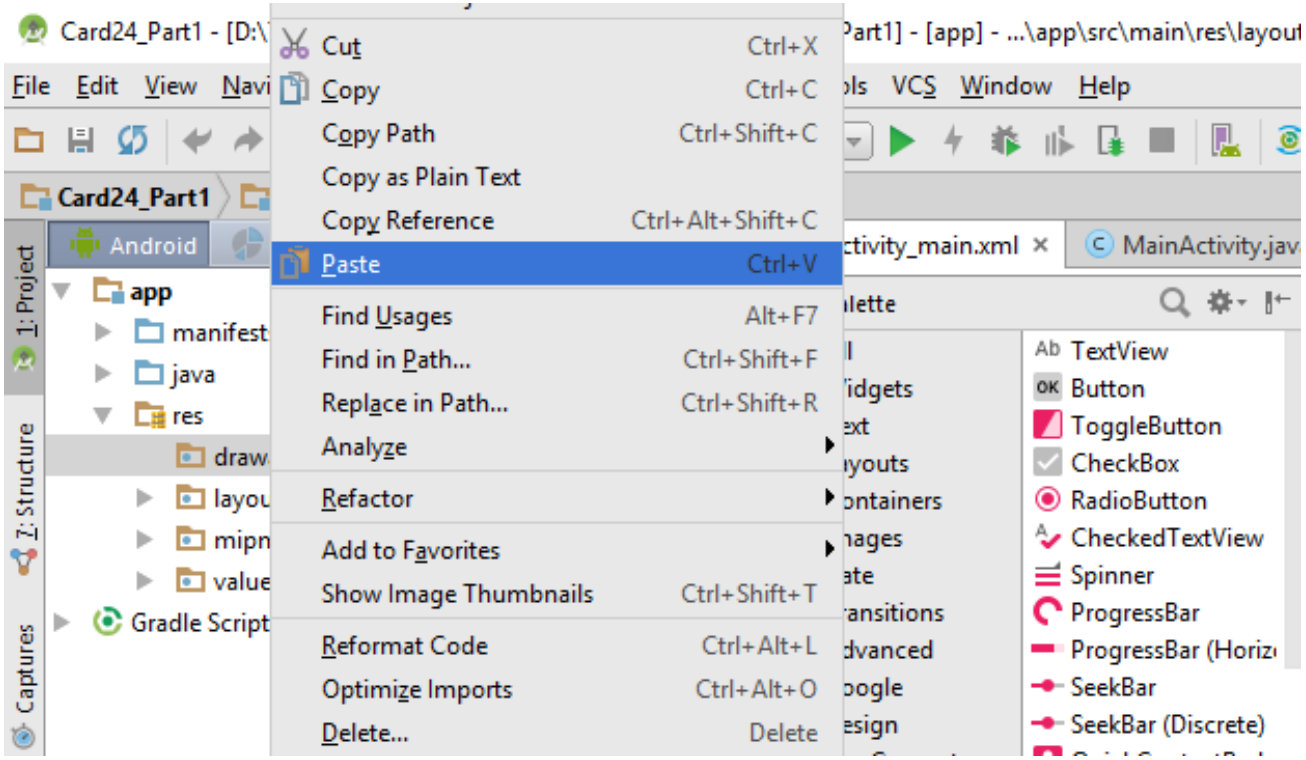
Task 2

- Replace the contents of “activity_main.xml” by that of “Layout_Template.txt”.
- Study the contents carefully.
- “Layout_Components.txt” contains some missing components. Copy them into proper locations of “activity_main.xml” so that your layout “Design” view is the same as the right diagram.



Task 3

- Copy the file “back_0.png” and paste it into the “drawable” folder of your project.



Task 4

- Press the green arrow button to compile and execute the program. You should not see any image for the cards.
- Let's include some simple codes to make them show the card back image.
- Open the main program "MainActivity.java" and do the following:
  - Define the following variable:
    - private lateinit var cards: Array<ImageButton> // lateinit: will be initialized later
  - Define the following method. Please replace "XXX" by your package name.
    - private fun initCardImage() {
      for (i in 0..3) {
      val resID: Int = resources.getIdentifier("back_0", "drawable", "hk.hkucs.card24")
      cards[i].setImageResource(resID)
      }
      }

Task 4

- Open the main program “MainActivity.java” and do the following:
  - Add the following to the “onCreate()” method.
    - cards = arrayOf(findViewById<ImageButton>(R.id.card1),
      findViewById<ImageButton>(R.id.card2),
      findViewById<ImageButton>(R.id.card3),
      findViewById<ImageButton>(R.id.card4))
  - Press the green arrow button to compile and execute the program.
  - Can you see the card back image? If not, please try to resolve the problem.

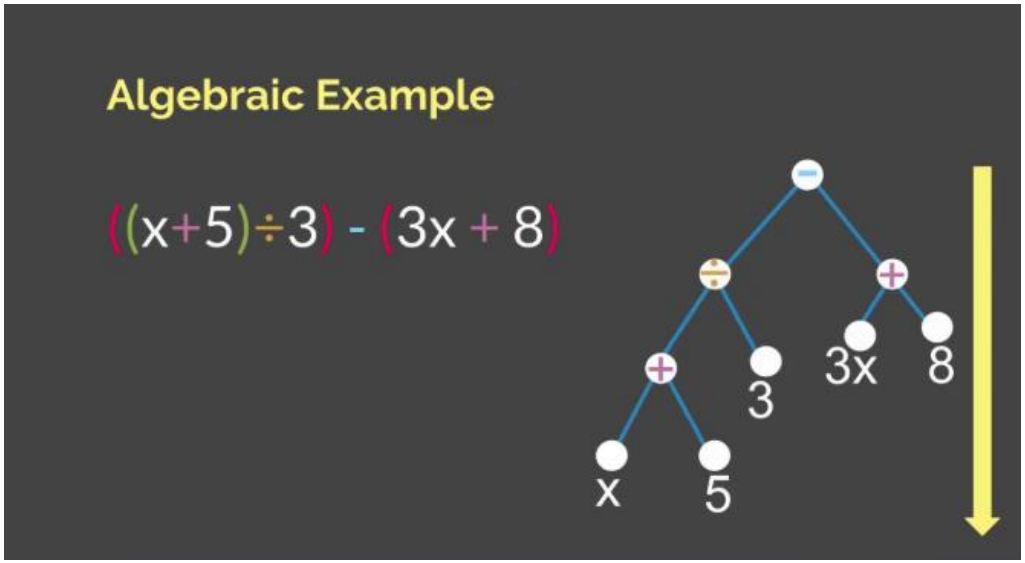
private lateinit var cards: Array<ImageButton>

lateinit-> i promise i will initialise the variable later

Sample Output

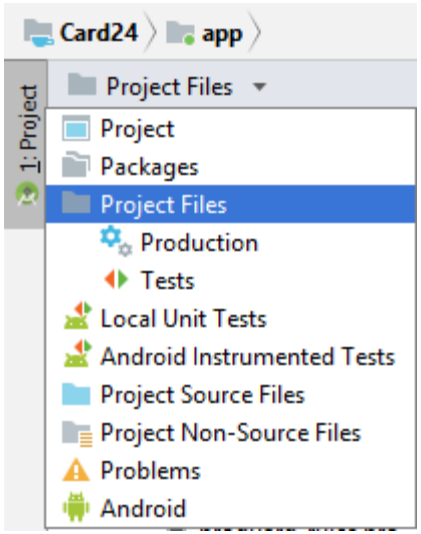
JEP Library

- The program involve the evaluations of expressions like "(13 – 5) * (1 + 2)".
- Since the evaluation orders of different parts of the expression are different, the implementation requires an Expression Tree, which is quite complicated.
- To simplify our job, we rely on the JEP Library.



Task 5: Adding JEP Library to Project

- Download JEP library (jep-java-3.4-trial.zip) from Moodle and extract the folder "jep-java-3.4-trial"
- Copy this folder into "app/libs" sub-folder of your project
- Go back to Android Studio. Go to "Project Files" tab and you should see the "libs/jep-java-3.4-trial" sub-folder in the "app" folder. Expand it and you should see the "jep-java-3.4-trial" folder. Inside this folder, you should see the file "jep-java-3.4-trial.jar" (so the full path is "app\libs\jep-java-3.4-trial\jep-java-3.4-trial\jep-java-3.4-trial.jar". Right-click this file and choose "Add As Library". Press "OK" to actually add the library to your project.



Task 5

- Add the following to the “onCreate()” method to test whether the imported JEP library works.

```
// For testing JEP library only
var jep = new Jep()
val res: Any = try {
                jep.parse("(3 + 3) * (2 + 2)")
                res = jep.evaluate()
} catch (e: ParseException) {
} catch (e: EvaluationException) {
}
val ca = res as Double
Toast.makeText(this@MainActivity, ca.toString(), Toast.LENGTH_LONG).show()
```

- Press the green arrow button to compile and execute the program.
- Can you see the pop-up box with the number “24.0” inside?

Save Your Work

- Please save and properly keep your work.
- Our next tutorial will base on your work today.
- You will be asked to submit your work some time after the next tutorial.

Tutorial 2.

End

2022-2023

COMP3330 Interactive Mobile Application Design and Programming

Dr. T.W. Chim (E-mail: twchim@cs.hku.hk)

Department of Computer Science, The University of Hong Kong