Tutorial 3.

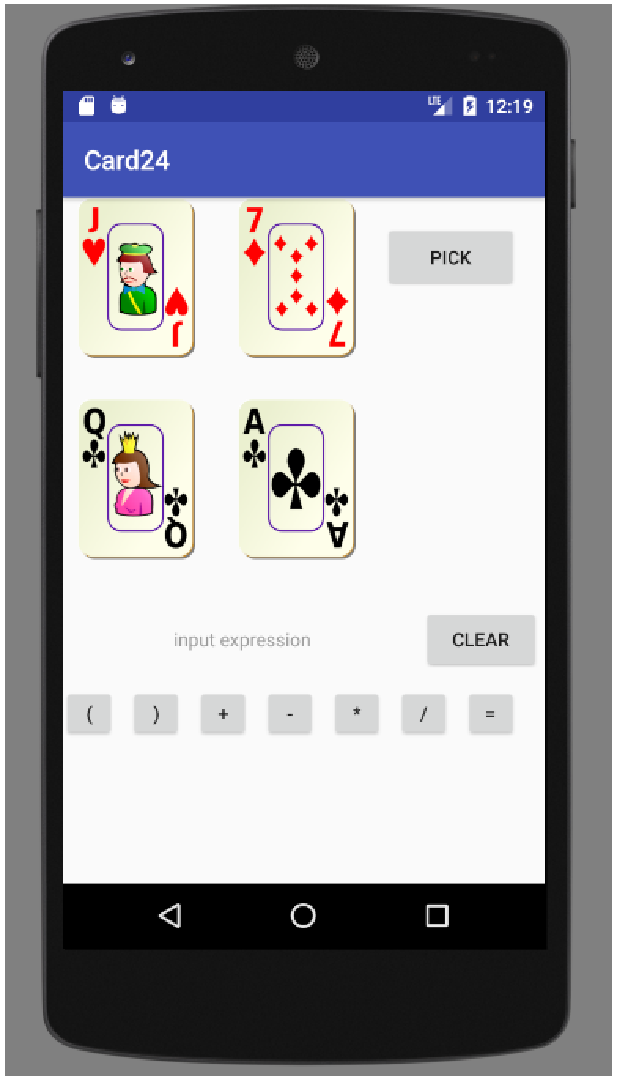
Card 24 Game (Part II)

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 tutorial, we will continue with the programming parts of the Android-based Card 24 game.
- Please open your work for Tutorial 2 in Android Studio.

Android Emulator - Nexus_5_API_23_x86:5554



Task 1: Link up components in layout files and main programs

- Open “MainActivity.java”. Define the following variables and add missing “import” statements where necessary.

```
private var rePick: Button? = null
private var checkInput: Button? = null
private var clear: Button? = null
private var left: Button? = null
private var right: Button? = null
private var plus: Button? = null
private var minus: Button? = null
private var multiply: Button? = null
private var divide: Button? = null
private lateinit var expression: TextView
private lateinit var cards: Array<ImageButton>   // 🡄 You should have
defined this last time!
```

Task 1: Link up components in layout files and main programs

- Link up these variables with the corresponding components in “activity_main.xml” layout file using “findViewById” statements in “onCreate” method:

```
rePick = findViewById<Button>(R.id.repick)
checkInput = findViewById<Button>(R.id.checkinput)
left = findViewById<Button>(R.id.left)
right = findViewById<Button>(R.id.right)
plus = findViewById<Button>(R.id.plus)
minus = findViewById<Button>(R.id.minus)
multiply = findViewById<Button>(R.id.multiply)
divide = findViewById<Button>(R.id.divide)
clear = findViewById<Button>(R.id.clear)
expression = findViewById<TextView>(R.id.input)
```

Press the green arrow button to compile and execute the program.

Task 2: Define a “pickCard” method for testing purpose

- Download “drawable.zip“ from Moodle and unzip it.
- Copy all images into the “drawable” folder of your Android Studio project.

Task 2: Define a "pickCard" method for testing purpose

- In "MainActivity.java", define the global variables "data", "card" and "imageCount" (for storing the values, the identifiers and the number of clicks of the four random cards) and the methods below:

```
private lateinit var data: Array<Int>
private lateinit var card: Array<Int>
private lateinit var imageCount: Array<Int>

private fun pickCard(){
        data = arrayOf(0, 0, 0, 0)
        card = arrayOf(0, 0, 0, 0)
        card[0] = 4
        card[1] = 5
        card[2] = 9
        card[3] = 10
        data[0] = 4
        data[1] = 5
        data[2] = 9
        data[3] = 10
        setClear()
}
```

Task 2: Define a “pickCard” method for testing purpose

```
private fun setClear() {
        var resID: Int
        expression.text = ""
        for (i in 0..3) {
                imageCount[i] = 0
                resID = resources.getIdentifier("card" + card[i], "drawable", "hk.hkucs.card24")
                cards[i].setImageResource(resID)
                cards[i].isClickable = true
        }
}
```

Task 2: Define a “pickCard” method for testing purpose

- Also instantiate imageCount in “onCreate()” method:

```
imageCount = arrayOf(0, 0, 0, 0)
```

- Call “pickCard()” in the “onCreate” method.
- Press the green arrow button to compile and execute the program.

Task 3: Define listeners for buttons

- In “MainActivity.java”, define a “clickCard” method. That is, when a card is clicked, this method will be invoked.

```
private fun clickCard(i: Int) {
        val resId: Int
        val num: String
        val value: Int
        if (imageCount[i] == 0) {
                resId = resources.getIdentifier("back_0", "drawable", "hk.hkucs.card24")
                cards[i].setImageResource(resId)
                cards[i].isClickable = false
                value = data[i]
                num = value.toString()
                expression.append(num)
                imageCount[i]++
        }
}
```

Task 3: Define listeners for buttons

- Then define a listener for “card[0]” in the “onCreate” method as follows:

```
cards[0].setOnClickListener(View.OnClickListener { clickCard(0) })
```

- Define listeners for “card[1]”, “card[2]” and “card[3]” in a similar manner.
- Add missing import statements when necessary.
- Then define a listener for “left” in the “onCreate” method as follows:

```
left!!.setOnClickListener { expression.append("(") }
```

- Define listeners for “right”, “plus”, “minus”, “multiply” and “divide” in a similar manner.

Task 3: Define listeners for buttons

- Then define a listener for “clear” in the “onCreate” method as follows:

```
clear!!.setOnClickListener { setClear() }
```

- Press the green arrow button to compile and execute the program.

Task 4: Define listener for “checkInput” button

- In “MainActivity.java”, define a “checkInput” method. That is, when the “checkInput” button is clicked, this method will be invoked to check whether the expression gives a result of 24. Here we make use of the external library JEP.

```
private fun checkInput(input: String): Boolean {
        val jep = Jep()
        val res: Any = try {
                jep.parse(input)
                jep.evaluate()
        } catch (e: ParseException) {
                e.printStackTrace()
                Toast.makeText(
                        this@MainActivity,
                        "Wrong Expression", Toast.LENGTH_SHORT
                ).show()
                return false
```

Task 4: Define listener for “checkInput” button

```
        } catch (e: EvaluationException) {
                e.printStackTrace()
                Toast.makeText(
                        this@MainActivity,
                        "Wrong Expression", Toast.LENGTH_SHORT
                ).show()
                return false
        }
        val ca = res as Double
        return abs(ca - 24) < 1e-6
}
```

Task 4: Define listener for “checkInput” button

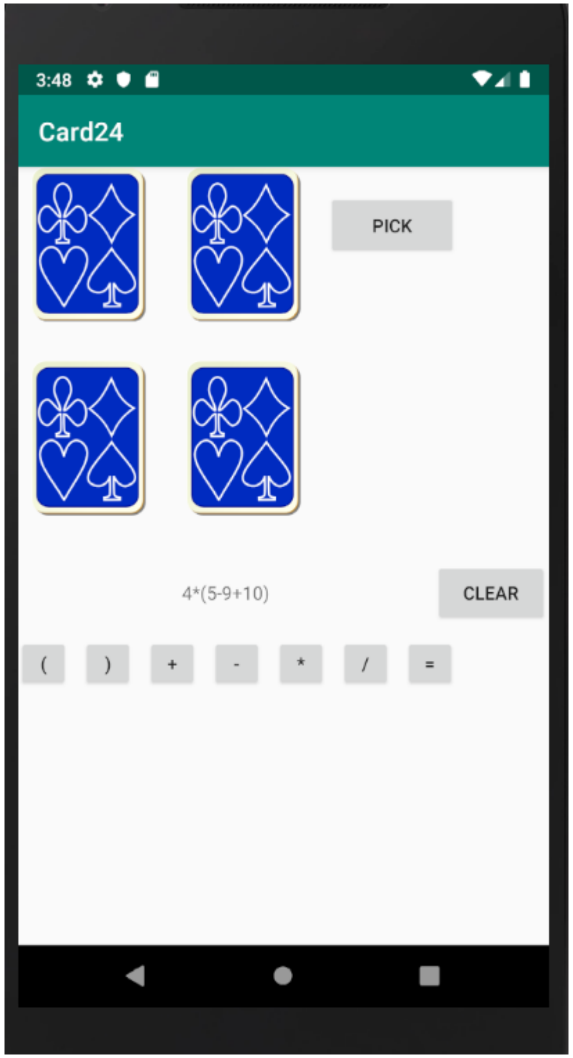
- Then define a listener for “checkInput” as follows:

```
checkInput!!.setOnClickListener {
        val inputStr: String = expression.text.toString()
        if (checkInput(inputStr)) {
                Toast.makeText(
                        this@MainActivity, "Correct Answer",
                        Toast.LENGTH_SHORT
                ).show()
                pickCard()
        } else {
                Toast.makeText(
                        this@MainActivity, "Wrong Answer",
                        Toast.LENGTH_SHORT
                ).show()
                setClear()
        }
}
```

- Press the green arrow button to compile and execute the program.

Testing

- Try to play around by entering your formula!
- If you cannot think of the answer, please try some online solver such as: http://scripts.cac.psu.edu/staff/r/j/rjg5/scripts/Math24.pl



Task 5: Additional Tasks…

- Now the Card 24 game is almost done. However, before the game is playable, some key components are still missing.
- Please complete the 2 tasks below.

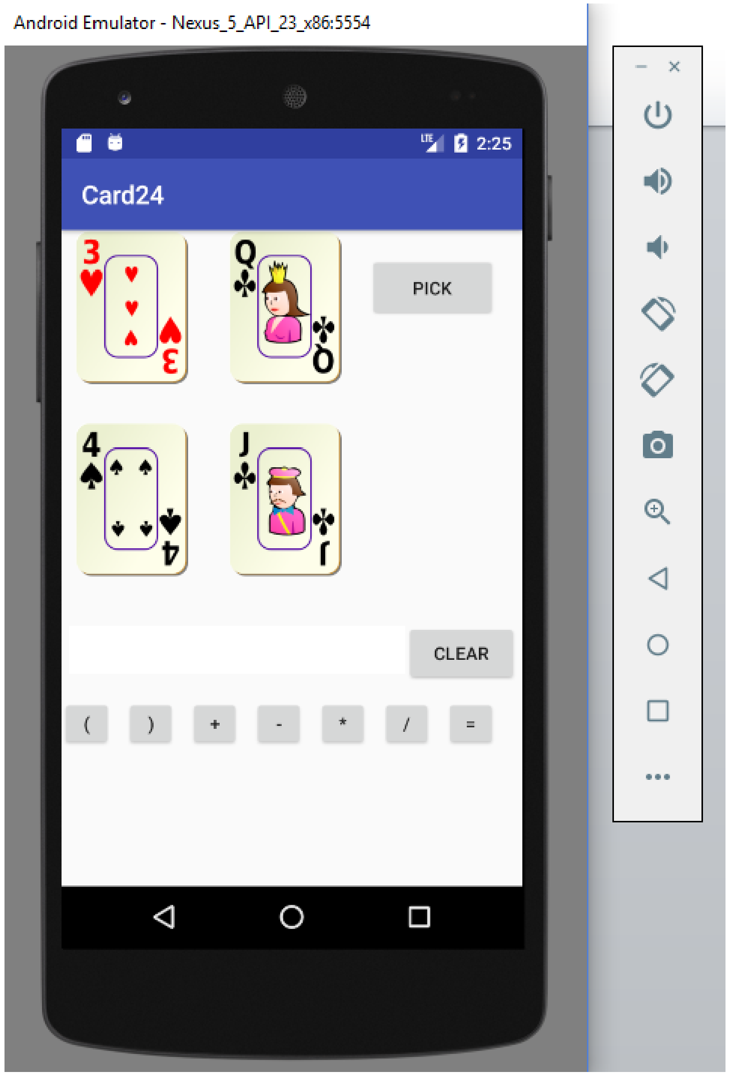
Task 5.1

- The poker card generation is not random. Recall that the four poker cards should be generated randomly.

Task 5.2

- The player should not choose fewer than four cards. Recall that in the traditional Card 24 game, all the four cards should be used.

Sample Run



"Wrong Answer" will be popped up if the result of your input expression is not equal to 24.

Save Your Work

- Please save your work, clean the project, zip the project folder and submit to Moodle by October 20, 2022 (Thursday) 23:59.

Tutorial 3.

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