Workshop 6 - 8

Card 24 Game (Android Version)

2024-2025

COMP7506 Smart Phone Apps Development

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

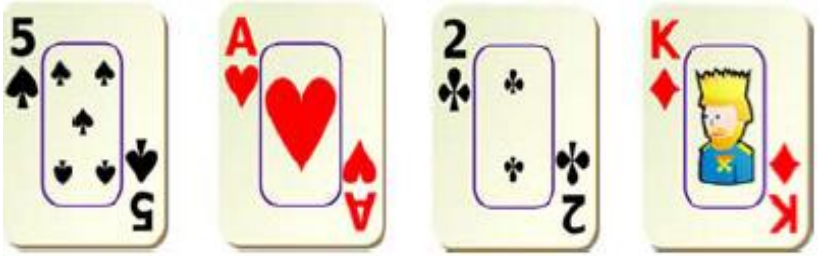
Department of Computer Science, School of Computing and Data Science,
Faculty of Engineering, The University of Hong Kong

Card 24 Game

- In these 3 workshops, you will learn how to build an Android-based Card 24 game.
- We will first focus on user interface design and then consider the 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 “Java” 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:

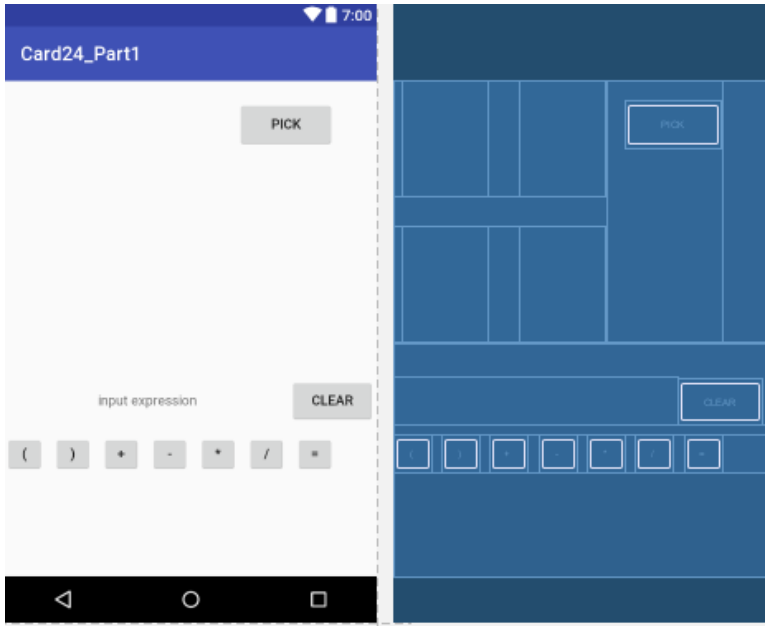
ScrollView

- 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>
```

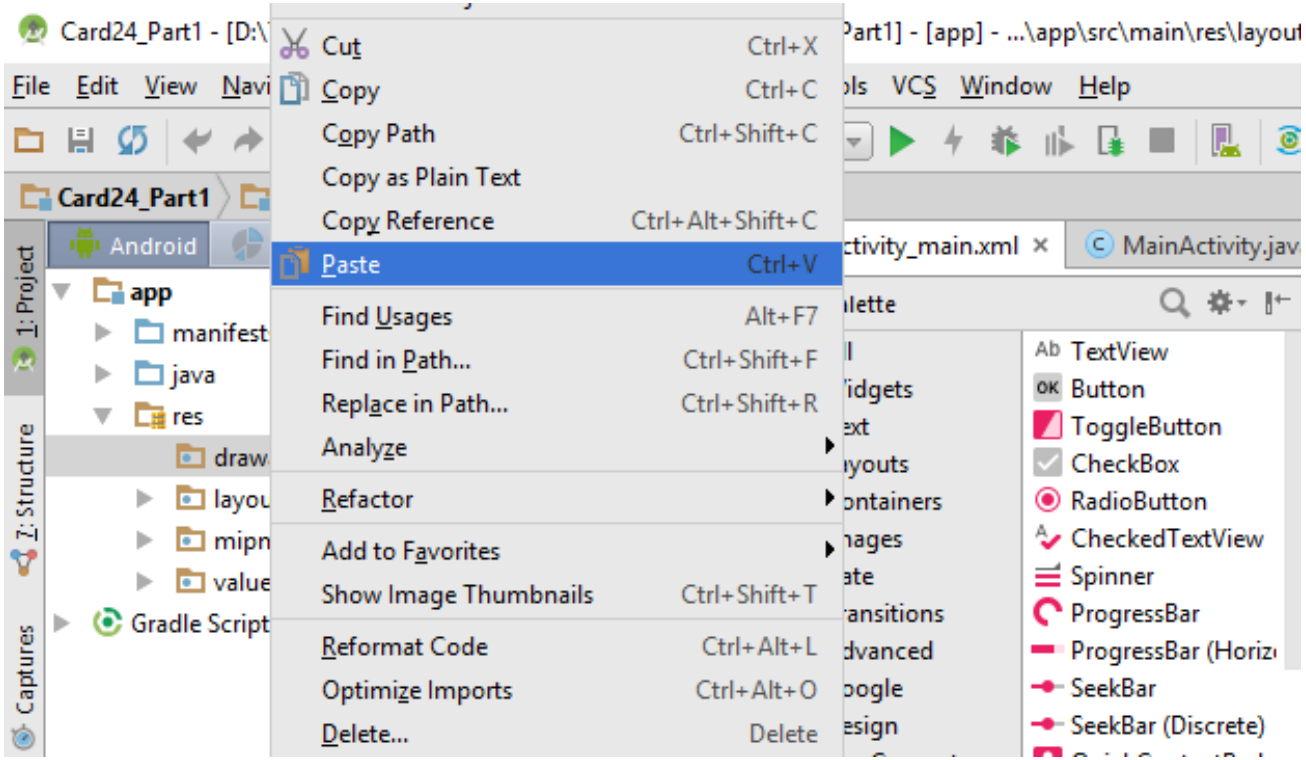
Task 2

- 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:
    - `ImageButton[] cards;`
  - Define the following method. Please replace "XXX" by your package name.

```
private void initCardImage() {
    for (int i = 0; i < 4; i++) {
        int resID =   getResources().getIdentifier("back_0","drawable",“XXX");
        cards[i].setImageResource(resID);
    }
}
```

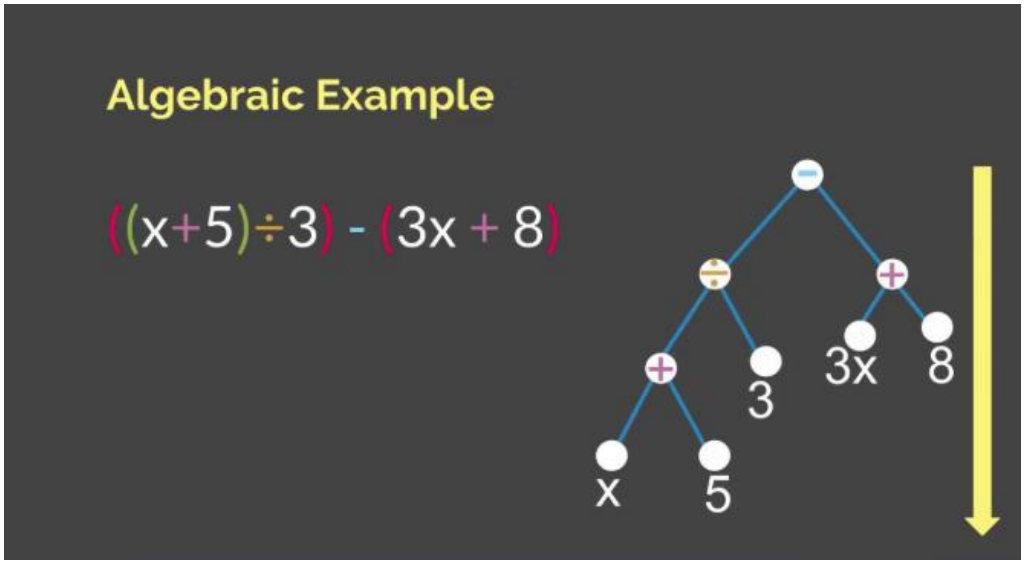
Task 4

- Open the main program “MainActivity.java” and do the following:
  - Add the following to the “onCreate()” method. Please replace “_____” by proper component names.
    - ```
      cards = new ImageButton[4];
      cards[0] = (ImageButton) findViewById(R.id._____);
      cards[1] = (ImageButton) findViewById(R.id._____);
      cards[2] = (ImageButton) findViewById(R.id._____);
      cards[3] = (ImageButton) findViewById(R.id._____);
      ```
  - 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.
      ```

Sample Output

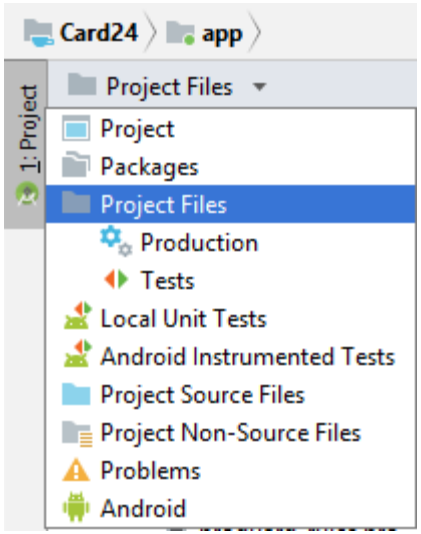
JEP Library

- 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
Jep jep = new Jep();
Object res = null;
try {
        jep.parse("(3 + 3) * (2 + 2)");
        res = jep.evaluate();
} catch (ParseException e) {
} catch (EvaluationException e) {
}
Double ca = (Double) res;
Toast.makeText(MainActivity.this, 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?

Task 6: Link up components in layout files and main programs

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

```
Button rePick;
Button checkInput;
Button clear;
Button left;
Button right;
Button plus;
Button minus;
Button multiply;
Button divide;
TextView expression;
ImageButton[] cards;   // ← You should have defined this last time!
```

Task 6: 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 = (Button)findViewById(R.id.repick);
checkInput = (Button)findViewById(R.id.checkinput);
left = (Button)findViewById(R.id.left);
right = (Button)findViewById(R.id.right);
plus = (Button)findViewById(R.id.plus);
minus = (Button)findViewById(R.id.minus);
multiply = (Button)findViewById(R.id.multiply);
divide = (Button)findViewById(R.id.divide);
clear = (Button)findViewById(R.id.clear);
expression = (TextView)findViewById(R.id.input);
```

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

Task 7: 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 7: 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:

```
int[] data;
int[] card;
int[] imageCount;

private void pickCard(){
        data = new int[4];
        card = new int[4];
        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 7: Define a “pickCard” method for testing purpose

```
private void setClear(){
        int resID;
        expression.setText("");
        for (int i = 0; i < 4; i++) {
                imageCount[i] = 0;
                resID = getResources().getIdentifier
                        ("card"+card[i],"drawable", “XXX");
                        // Please replace “XXX” by your package name
                cards[i].setImageResource(resID);
                cards[i].setClickable(true);
        }
}
```

Task 7: Define a “pickCard” method for testing purpose

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

```
imageCount = new int[4];
```

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

Task 8: Define listeners for buttons

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

```
private void clickCard(int i) {
        int resId;
        String num;
        Integer value;
        if (imageCount[i] == 0) {
                resId = getResources().getIdentifier("back_0","drawable", “XXX");
                // Please replace “XXX” by your package name
                cards[i].setImageResource(resId);
                cards[i].setClickable(false);
                value = data[i];
                num = value.toString();
                expression.append(num);
                imageCount[i] ++;
        }
}
```

Task 8: Define listeners for buttons

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

```
cards[0].setOnClickListener(new ImageButton.OnClickListener() {
        public void onClick(View view) {
                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(new Button.OnClickListener(){
        public void onClick(View view) {
                expression.append("(");
        }
});
```

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

Task 8: Define listeners for buttons

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

```
clear.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View view){
                setClear();
        }
});
```

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

Task 9: 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 boolean checkInput(String input) {
        Jep jep = new Jep();
        Object res;
        try {
                jep.parse(input);
                res = jep.evaluate();
        } catch (ParseException e) {
                e.printStackTrace();
                Toast.makeText(MainActivity.this,
                "Wrong Expression", Toast.LENGTH_SHORT).show();
                return false;
```

Task 9: Define listener for “checkInput” button

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

Task 9: Define listener for “checkInput” button

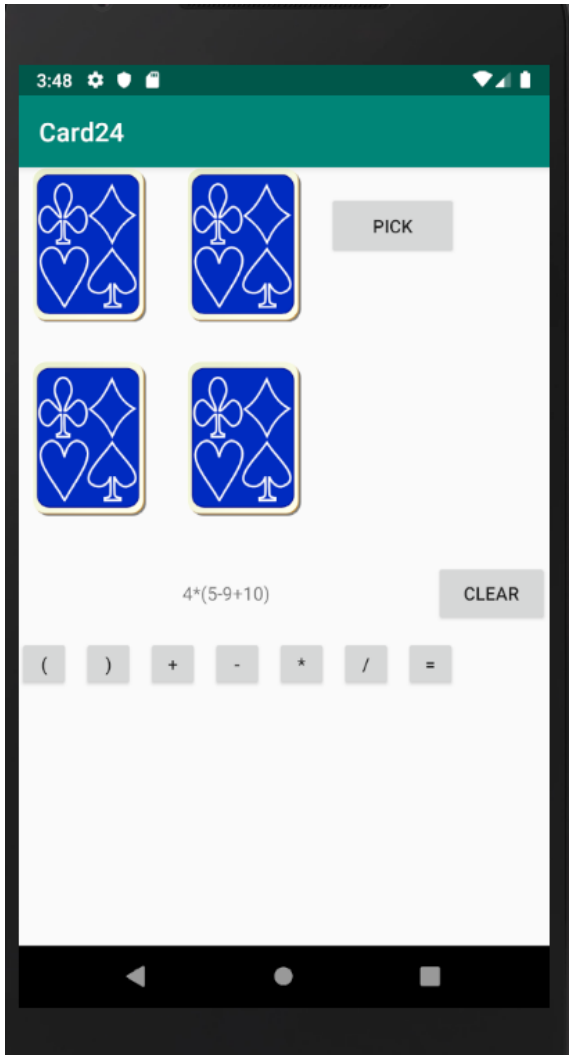
- Then define a listener for “checkInput” as follows:

```
checkInput.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View view) {
                String inputStr = expression.getText().toString();
                if (checkInput(inputStr)) {
                        Toast.makeText(MainActivity.this, "Correct Answer",
                        Toast.LENGTH_SHORT).show();
                        pickCard();
                } else {
                        Toast.makeText(MainActivity.this, "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://www.99cankao.com/numbers/24game.php



Task 10: 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 3 tasks below.

Task 10.1

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

Task 10.2

- The player can make non-sense input like “()1+/12”. Recall the limitations suggested earlier:
  - 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 “)”.

Task 10.3

- 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.

Workshop 6 - 8

End

2024-2025
COMP7506 Smart Phone Apps Development
Dr. T.W. Chim (E-mail: twchim@cs.hku.hk)
Department of Computer Science, School of Computing and Data Science,
Faculty of Engineering, The University of Hong Kong