Cours Programmation Mobile Avancée – Détail des TD

Durée totale : 30h | Format : TD/TP intégrés avec mini-projets

---

🎯 TD 1 : Introduction & Rappels Web (3h)

Objectifs pédagogiques

- Comprendre les différents paradigmes de développement mobile
- Maîtriser les fondamentaux du développement web responsive
- Identifier les spécificités du développement mobile

Contenu théorique (1h30)

1.1 Les paradigmes mobiles

- **Applications natives** : Performance optimale, accès complet aux API
  - iOS : Swift/Objective-C + Xcode
  - Android : Java/Kotlin + Android Studio
  - Avantages : UX native, performances, accès capteurs
  - Inconvénients : Coût de développement, maintenance double
- **Applications hybrides** : Une base de code, déploiement multiplateforme
  - Technologies : Ionic, React Native, Flutter, Xamarin
  - Avantages : ROI, compétences web réutilisables
  - Inconvénients : Performance moindre, dépendance frameworks
- **Progressive Web Apps (PWA)** : Web apps avec fonctionnalités natives
  - Service Workers, App Manifest, responsive design
  - Avantages : Mise à jour instantanée, pas d'app store
  - Limites : Accès limité aux API natives

1.2 Spécificités du développement mobile

- **Contraintes techniques** : Écrans tactiles, tailles variées, orientations
- **Ressources limitées** : CPU, RAM, batterie, connectivité
- **Paradigmes d'interaction** : Touch, gestures, voice
- **Contexte d'usage** : Mobilité, interruptions fréquentes

Travaux pratiques (1h30)

Exercice 1 : Analyse comparative (30min)

Analysez 3 applications populaires (ex: Instagram, Spotify, Uber) :

- Identifiez le type de développement (natif/hybride)
- Listez les fonctionnalités natives utilisées
- Évaluez l'expérience utilisateur

Exercice 2 : Page de login responsive (60min)

```html
```

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Mobile</title>
  <style>
    * { box-sizing: border-box; margin: 0; padding: 0; }

    body {
      font-family: 'Segoe UI', sans-serif;
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      padding: 20px;
    }

    .login-container {
      background: white;
      width: 100%;
      max-width: 400px;
      padding: 40px 30px;
      border-radius: 16px;
      box-shadow: 0 20px 40px rgba(0,0,0,0.1);
      animation: slideUp 0.5s ease;
    }

    @keyframes slideUp {
      from { transform: translateY(30px); opacity: 0; }
      to { transform: translateY(0); opacity: 1; }
    }

    h2 {
      text-align: center;
      margin-bottom: 30px;
      color: #333;
      font-size: 28px;
    }

    .input-group {
      margin-bottom: 20px;
      position: relative;
    }
```

```css
input {
    width: 100%;
    padding: 15px;
    border: 2px solid #e1e1e1;
    border-radius: 8px;
    font-size: 16px;
    transition: border-color 0.3s;
}

input:focus {
    outline: none;
    border-color: #667eea;
}

.login-btn {
    width: 100%;
    padding: 15px;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    border: none;
    border-radius: 8px;
    font-size: 16px;
    font-weight: 600;
    cursor: pointer;
    transition: transform 0.2s;
}

.login-btn:hover {
    transform: translateY(-2px);
}

.login-btn:active {
    transform: translateY(0);
}

.forgot-password {
    text-align: center;
    margin-top: 20px;
}

.forgot-password a {
    color: #667eea;
    text-decoration: none;
    font-size: 14px;
}

/* Media queries pour mobile */
```

```
        @media (max-width: 480px) {
            .login-container {
                padding: 30px 20px;
                margin: 10px;
            }

            h2 { font-size: 24px; }
            input, .login-btn { padding: 12px; }
        }
    </style>
</head>
<body>
    <div class="login-container">
        <h2>Connexion</h2>
        <form id="loginForm">
            <div class="input-group">
                <input type="email" id="email" placeholder="Email" required>
            </div>
            <div class="input-group">
                <input type="password" id="password" placeholder="Mot de passe" required>
            </div>
            <button type="submit" class="login-btn">Se connecter</button>
        </form>
        <div class="forgot-password">
            <a href="#">Mot de passe oublié ?</a>
        </div>
    </div>

    <script>
        document.getElementById('loginForm').addEventListener('submit', function(e) {
            e.preventDefault();
            const email = document.getElementById('email').value;
            const password = document.getElementById('password').value;

            // Validation basique
            if (email && password) {
                alert('Connexion simulée pour : ' + email);
            }
        });
    </script>
</body>
</html>
```

À retenir : Design mobile-first, animations subtiles, validation client

---

🚀 TD 2 : Ionic & Cordova (4h)

Objectifs pédagogiques

- Maîtriser l'architecture Ionic + Angular
- Comprendre le rôle de Cordova/PhoneGap
- Développer une première application hybride

Contenu théorique (1h)

2.1 Architecture Ionic

- **Stack technologique** : Angular + Ionic UI + Cordova/Capacitor
- **Composants Ionic** : ion-button, ion-input, ion-list, ion-nav
- **Theming** : Variables CSS, modes iOS/Android
- **Navigation** : Router Angular, ion-nav, ion-tabs

2.2 Cordova vs Capacitor

- **Cordova** : Wrapper WebView historique, plugins JavaScript
- **Capacitor** : Nouvelle génération, meilleure intégration native
- **Build process** : ionic build → cordova prepare → compile native

Travaux pratiques (3h)

Application Todo-list complète

Phase 1 : Setup & structure (45min)

```bash
npm install -g @ionic/cli
ionic start todoApp blank --type=angular
cd todoApp
ionic serve
```

Phase 2 : Modèle de données (30min)

```typescript
```

```typescript
// models/task.model.ts
export interface Task {
  id: number;
  title: string;
  description?: string;
  completed: boolean;
  priority: 'low' | 'medium' | 'high';
  createdAt: Date;
  dueDate?: Date;
}
```

Phase 3 : Service de gestion (45min)

typescript

```typescript
// services/task.service.ts
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';
import { Task } from '../models/task.model';

@Injectable({
  providedIn: 'root'
})
export class TaskService {
  private tasks: Task[] = [];
  private tasksSubject = new BehaviorSubject<Task[]>([]);
  public tasks$ = this.tasksSubject.asObservable();
  private nextId = 1;

  addTask(title: string, description?: string, priority: 'low' | 'medium' | 'high' = 'medium') {
    const newTask: Task = {
      id: this.nextId++,
      title: title.trim(),
      description: description?.trim(),
      completed: false,
      priority,
      createdAt: new Date()
    };

    this.tasks.unshift(newTask);
    this.tasksSubject.next([...this.tasks]);
  }

  toggleTask(id: number) {
    const task = this.tasks.find(t => t.id === id);
    if (task) {
      task.completed = !task.completed;
      this.tasksSubject.next([...this.tasks]);
    }
  }

  deleteTask(id: number) {
    this.tasks = this.tasks.filter(t => t.id !== id);
    this.tasksSubject.next([...this.tasks]);
  }

  getTasksByPriority() {
    return this.tasks.sort((a, b) => {
      const priorityOrder = { high: 3, medium: 2, low: 1 };
      return priorityOrder[b.priority] - priorityOrder[a.priority];
    });
```

```
  }
}
```

Phase 4 : Interface utilisateur (60min)

```html
```

```
<!-- home.page.html -->
<ion-header [translucent]="true">
  <ion-toolbar color="primary">
    <ion-title>Ma Todo-List</ion-title>
    <ion-buttons slot="end">
      <ion-badge color="light">{{ (tasks$ | async)?.length || 0 }}</ion-badge>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  <!-- Formulaire d'ajout -->
  <ion-card>
    <ion-card-content>
      <ion-item>
        <ion-input
          [(ngModel)]="newTask"
          placeholder="Nouvelle tâche..."
          (keyup.enter)="addTask()"
          clearInput="true">
        </ion-input>
      </ion-item>

      <ion-item>
        <ion-select [(ngModel)]="selectedPriority" placeholder="Priorité">
          <ion-select-option value="low">Basse</ion-select-option>
          <ion-select-option value="medium">Moyenne</ion-select-option>
          <ion-select-option value="high">Haute</ion-select-option>
        </ion-select>
      </ion-item>

      <ion-button expand="block" (click)="addTask()" [disabled]="!newTask?.trim()">
        <ion-icon name="add" slot="start"></ion-icon>
        Ajouter la tâche
      </ion-button>
    </ion-card-content>
  </ion-card>

  <!-- Liste des tâches -->
  <div *ngIf="(tasks$ | async)?.length === 0" class="empty-state">
    <ion-icon name="checkmark-circle-outline" size="large"></ion-icon>
    <p>Aucune tâche à afficher</p>
  </div>

  <ion-list *ngIf="(tasks$ | async)?.length > 0">
    <ion-item-sliding *ngFor="let task of tasks$ | async; trackBy: trackByFn">
```

```
      <ion-item
        [class.completed]="task.completed"
        [color]="task.priority === 'high' ? 'danger' : task.priority === 'medium' ? 'warning' : 'medium'">

        <ion-checkbox
          slot="start"
          [(ngModel)]="task.completed"
          (ionChange)="toggleTask(task.id)">
        </ion-checkbox>

        <ion-label>
          <h2 [class.strikethrough]="task.completed">{{ task.title }}</h2>
          <p *ngIf="task.description">{{ task.description }}</p>
          <p class="task-date">{{ task.createdAt | date:'dd/MM/yyyy HH:mm' }}</p>
        </ion-label>

        <ion-badge slot="end" [color]="getPriorityColor(task.priority)">
          {{ task.priority }}
        </ion-badge>
      </ion-item>

      <ion-item-options side="end">
        <ion-item-option color="danger" (click)="deleteTask(task.id)">
          <ion-icon name="trash" slot="icon-only"></ion-icon>
        </ion-item-option>
      </ion-item-options>
    </ion-item-sliding>
  </ion-list>
</ion-content>
```

CSS personnalisé :

```
scss
```

```scss
// home.page.scss
.completed {
  opacity: 0.6;
}

.strikethrough {
  text-decoration: line-through;
}

.empty-state {
  text-align: center;
  padding: 60px 20px;
  color: var(--ion-color-medium);

  ion-icon {
    margin-bottom: 20px;
  }
}

.task-date {
  font-size: 12px;
  color: var(--ion-color-medium);
}
```

À retenir : Architecture MVVM, observables RxJS, composants Ionic

---

📷 TD 3 : Plugins Natifs (4h)

Objectifs pédagogiques

- Intégrer des plugins Cordova/Capacitor
- Accéder aux fonctionnalités natives (caméra, GPS, stockage)
- Gérer les permissions et la sécurité

Contenu théorique (1h)

3.1 Écosystème des plugins

- **Plugins officiels** : Camera, Geolocation, Device, File
- **Plugins communautaires** : Awesome Cordova Plugins
- **Capacitor vs Cordova** : Différences d'API et de performance

3.2 Gestion des permissions

- **Android** : Manifest + demandes runtime
- **iOS** : Info.plist + Privacy descriptions
- **Bonnes pratiques** : Expliciter l'utilisation, graceful degradation

Travaux pratiques (3h)

Application Photo Gallery avec fonctionnalités avancées

Phase 1 : Installation des plugins (20min)

```bash
ionic capacitor plugin add @capacitor/camera
ionic capacitor plugin add @capacitor/filesystem
ionic capacitor plugin add @capacitor/preferences
npm install @capacitor/camera @capacitor/filesystem @capacitor/preferences
ionic capacitor sync
```

Phase 2 : Service photo (60min)

```typescript
```

```
// services/photo.service.ts
import { Injectable } from '@angular/core';
import { Camera, CameraResultType, CameraSource, Photo } from '@capacitor/camera';
import { Filesystem, Directory } from '@capacitor/filesystem';
import { Preferences } from '@capacitor/preferences';

export interface UserPhoto {
  filepath: string;
  webviewPath?: string;
  base64?: string;
  name: string;
  date: string;
  size?: number;
}

@Injectable({
  providedIn: 'root'
})
export class PhotoService {
  public photos: UserPhoto[] = [];
  private PHOTO_STORAGE = 'photos';

  constructor() {
    this.loadSaved();
  }

  public async addNewToGallery() {
    try {
      const capturedPhoto = await Camera.getPhoto({
        resultType: CameraResultType.Uri,
        source: CameraSource.Camera,
        quality: 80,
        width: 800,
        height: 600,
        allowEditing: false
      });

      const savedImageFile = await this.savePicture(capturedPhoto);
      this.photos.unshift(savedImageFile);

      await Preferences.set({
        key: this.PHOTO_STORAGE,
        value: JSON.stringify(this.photos)
      });

      return savedImageFile;
```

```
  } catch (error) {
    console.error('Erreur lors de la prise de photo:', error);
    throw error;
  }
}

public async selectFromGallery() {
  try {
    const selectedPhoto = await Camera.getPhoto({
      resultType: CameraResultType.Uri,
      source: CameraSource.Photos,
      quality: 80
    });

    const savedImageFile = await this.savePicture(selectedPhoto);
    this.photos.unshift(savedImageFile);

    await this.savePhotosData();
    return savedImageFile;
  } catch (error) {
    console.error('Erreur lors de la sélection:', error);
    throw error;
  }
}

private async savePicture(photo: Photo) {
  const base64Data = await this.readAsBase64(photo);
  const fileName = `photo_${new Date().getTime()}.jpeg`;

  const savedFile = await Filesystem.writeFile({
    path: fileName,
    data: base64Data,
    directory: Directory.Data
  });

  return {
    filepath: savedFile.uri,
    webviewPath: photo.webPath,
    base64: base64Data,
    name: fileName,
    date: new Date().toISOString()
  } as UserPhoto;
}

private async readAsBase64(photo: Photo) {
  const response = await fetch(photo.webPath!);
  const blob = await response.blob();
```

```
  return await this.convertBlobToBase64(blob) as string;
}

private convertBlobToBase64 = (blob: Blob) => new Promise((resolve, reject) => {
  const reader = new FileReader();
  reader.onerror = reject;
  reader.onload = () => {
    resolve(reader.result);
  };
  reader.readAsDataURL(blob);
});

public async deletePhoto(photo: UserPhoto, position: number) {
  this.photos.splice(position, 1);

  await Preferences.set({
    key: this.PHOTO_STORAGE,
    value: JSON.stringify(this.photos)
  });

  try {
    await Filesystem.deleteFile({
      path: photo.name,
      directory: Directory.Data
    });
  } catch (error) {
    console.error('Erreur lors de la suppression:', error);
  }
}

private async loadSaved() {
  const { value } = await Preferences.get({ key: this.PHOTO_STORAGE });
  this.photos = (value ? JSON.parse(value) : []) as UserPhoto[];

  // Mise à jour des chemins web pour l'affichage
  for (let photo of this.photos) {
    const readFile = await Filesystem.readFile({
      path: photo.name,
      directory: Directory.Data
    });
    photo.webviewPath = `${readFile.data}`;
  }
}

private async savePhotosData() {
  await Preferences.set({
```

```
      key: this.PHOTO_STORAGE,
      value: JSON.stringify(this.photos)
    });
  }
}
```

Phase 3 : Interface gallery (80min)

```
html
```

```
<!-- gallery.page.html -->
<ion-header>
  <ion-toolbar color="primary">
    <ion-title>Ma Galerie</ion-title>
    <ion-buttons slot="end">
      <ion-button (click)="presentActionSheet()">
        <ion-icon name="add-circle" slot="icon-only"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div class="gallery-info" *ngIf="photoService.photos.length > 0">
    <p>{{ photoService.photos.length }} photo(s) dans votre galerie</p>
  </div>

  <div class="empty-gallery" *ngIf="photoService.photos.length === 0">
    <ion-icon name="camera-outline" size="large"></ion-icon>
    <h2>Galerie vide</h2>
    <p>Ajoutez votre première photo !</p>
    <ion-button expand="block" (click)="presentActionSheet()">
      <ion-icon name="camera" slot="start"></ion-icon>
      Ajouter une photo
    </ion-button>
  </div>

  <div class="photo-grid">
    <div class="photo-container"
        *ngFor="let photo of photoService.photos; index as position"
        (click)="showPhotoDetails(photo, position)">

      <img [src]="photo.webviewPath" [alt]="photo.name">

      <div class="photo-overlay">
        <div class="photo-info">
          <p class="photo-name">{{ photo.name }}</p>
          <p class="photo-date">{{ photo.date | date:'dd/MM/yyyy HH:mm' }}</p>
        </div>

        <ion-button
          fill="clear"
          size="small"
          color="danger"
          (click)="confirmDelete(photo, position); $event.stopPropagation()">
          <ion-icon name="trash" slot="icon-only"></ion-icon>
```

```
      </ion-button>
    </div>
  </div>
 </div>

 <!-- FAB pour ajouter rapidement -->
 <ion-fab vertical="bottom" horizontal="end" slot="fixed">
  <ion-fab-button (click)="addPhotoToGallery()">
   <ion-icon name="camera"></ion-icon>
  </ion-fab-button>
 </ion-fab>
</ion-content>
```

À retenir : Gestion asynchrone, stockage local, permissions natives

---

☕ TD 4 : Android Natif (Java) (4h)

Objectifs pédagogiques

- Découvrir Android Studio et la structure Android
- Programmer en Java pour Android
- Comprendre le cycle de vie des Activities

Contenu théorique (1h)

4.1 Architecture Android

- **Composants** : Activities, Services, BroadcastReceivers, ContentProviders
- **Cycle de vie** : onCreate, onStart, onResume, onPause, onStop, onDestroy
- **Layout system** : LinearLayout, RelativeLayout, ConstraintLayout
- **Resources** : strings.xml, colors.xml, dimens.xml

4.2 Bonnes pratiques Java Android

- **Null safety** : Annotations @Nullable/@NonNull
- **Memory leaks** : Éviter les références statiques aux Context
- **Threading** : UI Thread vs Background threads

Travaux pratiques (3h)

Calculatrice scientifique avancée

Structure du projet :

```
app/
├── src/main/java/com/example/calculator/
│   ├── MainActivity.java
│   ├── CalculatorEngine.java
│   └── HistoryActivity.java
├── res/
│   ├── layout/
│   │   ├── activity_main.xml
│   │   └── activity_history.xml
│   ├── values/
│   │   ├── strings.xml
│   │   ├── colors.xml
│   │   └── styles.xml
```

MainActivity.java (2h)

java

```java
package com.example.calculator;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private TextView displayText;
    private TextView resultText;
    private CalculatorEngine engine;
    private ArrayList<String> history;
    private boolean isNewCalculation = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        initializeViews();
        initializeEngine();
        setupButtonListeners();
    }

    private void initializeViews() {
        displayText = findViewById(R.id.displayText);
        resultText = findViewById(R.id.resultText);
        history = new ArrayList<>();
    }

    private void initializeEngine() {
        engine = new CalculatorEngine();
    }

    private void setupButtonListeners() {
        // Chiffres 0-9
        int[] numberIds = {R.id.btn0, R.id.btn1, R.id.btn2, R.id.btn3, R.id.btn4,
                           R.id.btn5, R.id.btn6, R.id.btn7, R.id.btn8, R.id.btn9};

        for (int i = 0; i < numberIds.length; i++) {
            final int number = i;
```

```
            findViewById(numberIds[i]).setOnClickListener(v -> appendNumber(String.valueOf(number)));
        }

        // Opérateurs
        findViewById(R.id.btnAdd).setOnClickListener(v -> appendOperator("+"));
        findViewById(R.id.btnSubtract).setOnClickListener(v -> appendOperator("-"));
        findViewById(R.id.btnMultiply).setOnClickListener(v -> appendOperator("×"));
        findViewById(R.id.btnDivide).setOnClickListener(v -> appendOperator("÷"));

        // Fonctions spéciales
        findViewById(R.id.btnEquals).setOnClickListener(v -> calculateResult());
        findViewById(R.id.btnClear).setOnClickListener(v -> clearAll());
        findViewById(R.id.btnDelete).setOnClickListener(v -> deleteLastChar());
        findViewById(R.id.btnDecimal).setOnClickListener(v -> appendDecimal());

        // Fonctions scientifiques
        findViewById(R.id.btnSin).setOnClickListener(v -> appendFunction("sin("));
        findViewById(R.id.btnCos).setOnClickListener(v -> appendFunction("cos("));
        findViewById(R.id.btnTan).setOnClickListener(v -> appendFunction("tan("));
        findViewById(R.id.btnLog).setOnClickListener(v -> appendFunction("log("));
        findViewById(R.id.btnSqrt).setOnClickListener(v -> appendFunction("√("));
        findViewById(R.id.btnPower).setOnClickListener(v -> appendOperator("^"));

        // Historique
        findViewById(R.id.btnHistory).setOnClickListener(v -> openHistory());
    }

    private void appendNumber(String number) {
        if (isNewCalculation) {
            displayText.setText(number);
            isNewCalculation = false;
        } else {
            String currentText = displayText.getText().toString();
            if (!currentText.equals("0")) {
                displayText.setText(currentText + number);
            } else {
                displayText.setText(number);
            }
        }
        updateResult();
    }

    private void openHistory() {
        Intent intent = new Intent(this, HistoryActivity.class);
        intent.putStringArrayListExtra("history", history);
        startActivity(intent);
    }
```

```
    private boolean isOperator(char c) {
        return c == '+' || c == '-' || c == '×' || c == '÷' || c == '^';
    }
}
```

CalculatorEngine.java (45min)

java

```java
package com.example.calculator;

import java.util.Stack;
import java.util.StringTokenizer;

public class CalculatorEngine {

    public double evaluate(String expression) throws Exception {
        // Préprocessing pour gérer les fonctions
        expression = preprocessExpression(expression);

        // Utilisation de l'algorithme Shunting Yard pour conversion en notation polonaise inversée
        return evaluateRPN(infixToRPN(expression));
    }

    private String preprocessExpression(String expression) {
        expression = expression.replace("×", "*").replace("÷", "/");
        expression = expression.replace("√", "sqrt");

        // Gérer les fonctions trigonométriques
        expression = expression.replace("sin(", "sin ");
        expression = expression.replace("cos(", "cos ");
        expression = expression.replace("tan(", "tan ");
        expression = expression.replace("log(", "log ");
        expression = expression.replace("sqrt(", "sqrt ");

        return expression;
    }

    private String infixToRPN(String expression) throws Exception {
        StringBuilder output = new StringBuilder();
        Stack<String> operators = new Stack<>();
        StringTokenizer tokenizer = new StringTokenizer(expression, "+-*/^() ", true);

        while (tokenizer.hasMoreTokens()) {
            String token = tokenizer.nextToken().trim();
            if (token.isEmpty()) continue;

            if (isNumber(token)) {
                output.append(token).append(" ");
            } else if (isFunction(token)) {
                operators.push(token);
            } else if (token.equals("(")) {
                operators.push(token);
            } else if (token.equals(")")) {
                while (!operators.isEmpty() && !operators.peek().equals("(")) {
```

```
                output.append(operators.pop()).append(" ");
            }
            if (!operators.isEmpty() && operators.peek().equals("(")) {
                operators.pop(); // Remove '('
            }
            if (!operators.isEmpty() && isFunction(operators.peek())) {
                output.append(operators.pop()).append(" ");
            }
        } else if (isOperator(token)) {
            while (!operators.isEmpty() &&
                   !operators.peek().equals("(") &&
                   getPrecedence(operators.peek()) >= getPrecedence(token)) {
                output.append(operators.pop()).append(" ");
            }
            operators.push(token);
        }
    }

    while (!operators.isEmpty()) {
        output.append(operators.pop()).append(" ");
    }

    return output.toString().trim();
}

private double evaluateRPN(String rpn) throws Exception {
    Stack<Double> operands = new Stack<>();
    StringTokenizer tokenizer = new StringTokenizer(rpn);

    while (tokenizer.hasMoreTokens()) {
        String token = tokenizer.nextToken();

        if (isNumber(token)) {
            operands.push(Double.parseDouble(token));
        } else if (isFunction(token)) {
            if (operands.isEmpty()) throw new Exception("Invalid expression");
            double operand = operands.pop();
            operands.push(applyFunction(token, operand));
        } else if (isOperator(token)) {
            if (operands.size() < 2) throw new Exception("Invalid expression");
            double b = operands.pop();
            double a = operands.pop();
            operands.push(applyOperator(token, a, b));
        }
    }

    if (operands.size() != 1) throw new Exception("Invalid expression");
```

```
        return operands.pop();
    }

    private double applyFunction(String function, double operand) throws Exception {
        switch (function) {
            case "sin": return Math.sin(Math.toRadians(operand));
            case "cos": return Math.cos(Math.toRadians(operand));
            case "tan": return Math.tan(Math.toRadians(operand));
            case "log": return Math.log10(operand);
            case "sqrt": return Math.sqrt(operand);
            default: throw new Exception("Unknown function: " + function);
        }
    }

    private double applyOperator(String operator, double a, double b) throws Exception {
        switch (operator) {
            case "+": return a + b;
            case "-": return a - b;
            case "*": return a * b;
            case "/":
                if (b == 0) throw new Exception("Division by zero");
                return a / b;
            case "^": return Math.pow(a, b);
            default: throw new Exception("Unknown operator: " + operator);
        }
    }

    private boolean isNumber(String token) {
        try {
            Double.parseDouble(token);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }

    private boolean isOperator(String token) {
        return "+-*/^".contains(token);
    }

    private boolean isFunction(String token) {
        return token.equals("sin") || token.equals("cos") || token.equals("tan") ||
               token.equals("log") || token.equals("sqrt");
    }

    private int getPrecedence(String operator) {
        switch (operator) {
```

```
            case "+":
            case "-": return 1;
            case "*":
            case "/": return 2;
            case "^": return 3;
            default: return 0;
        }
    }
}
```

Layout activity_main.xml (30min)

xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/background_primary">

    <!-- Zone d'affichage -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:gravity="bottom"
        android:padding="16dp"
        android:background="@color/display_background">

        <TextView
            android:id="@+id/displayText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="0"
            android:textSize="32sp"
            android:textColor="@color/display_text"
            android:gravity="end"
            android:fontFamily="monospace"
            android:layout_marginBottom="8dp" />

        <TextView
            android:id="@+id/resultText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textColor="@color/result_text"
            android:gravity="end"
            android:fontFamily="monospace" />

    </LinearLayout>

    <!-- Boutons fonctions scientifiques -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="8dp">
```

```
    <Button
        android:id="@+id/btnSin"
        style="@style/ScientificButton"
        android:text="sin" />

    <Button
        android:id="@+id/btnCos"
        style="@style/ScientificButton"
        android:text="cos" />

    <Button
        android:id="@+id/btnTan"
        style="@style/ScientificButton"
        android:text="tan" />

    <Button
        android:id="@+id/btnLog"
        style="@style/ScientificButton"
        android:text="log" />

    <Button
        android:id="@+id/btnSqrt"
        style="@style/ScientificButton"
        android:text="√" />

</LinearLayout>

<!-- Ligne 1: Clear, Delete, Power, Divide -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/btnClear"
        style="@style/OperatorButton"
        android:text="C"
        android:backgroundTint="@color/clear_button" />

    <Button
        android:id="@+id/btnDelete"
        style="@style/OperatorButton"
        android:text="⌫" />

    <Button
        android:id="@+id/btnPower"
        style="@style/OperatorButton"
```

```
        android:text="^" />

    <Button
        android:id="@+id/btnDivide"
        style="@style/OperatorButton"
        android:text="÷" />

</LinearLayout>

<!-- Ligne 2: 7, 8, 9, Multiply -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/btn7"
        style="@style/NumberButton"
        android:text="7" />

    <Button
        android:id="@+id/btn8"
        style="@style/NumberButton"
        android:text="8" />

    <Button
        android:id="@+id/btn9"
        style="@style/NumberButton"
        android:text="9" />

    <Button
        android:id="@+id/btnMultiply"
        style="@style/OperatorButton"
        android:text="×" />

</LinearLayout>

<!-- Ligne 3: 4, 5, 6, Subtract -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/btn4"
        style="@style/NumberButton"
        android:text="4" />
```

```
    <Button
        android:id="@+id/btn5"
        style="@style/NumberButton"
        android:text="5" />

    <Button
        android:id="@+id/btn6"
        style="@style/NumberButton"
        android:text="6" />

    <Button
        android:id="@+id/btnSubtract"
        style="@style/OperatorButton"
        android:text="-" />

</LinearLayout>

<!-- Ligne 4: 1, 2, 3, Add -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/btn1"
        style="@style/NumberButton"
        android:text="1" />

    <Button
        android:id="@+id/btn2"
        style="@style/NumberButton"
        android:text="2" />

    <Button
        android:id="@+id/btn3"
        style="@style/NumberButton"
        android:text="3" />

    <Button
        android:id="@+id/btnAdd"
        style="@style/OperatorButton"
        android:text="+" />

</LinearLayout>

<!-- Ligne 5: History, 0, Decimal, Equals -->
```

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btnHistory"
            style="@style/OperatorButton"
            android:text="📊" />

        <Button
            android:id="@+id/btn0"
            style="@style/NumberButton"
            android:text="0" />

        <Button
            android:id="@+id/btnDecimal"
            style="@style/NumberButton"
            android:text="." />

        <Button
            android:id="@+id/btnEquals"
            style="@style/EqualsButton"
            android:text="=" />

    </LinearLayout>

</LinearLayout>
```

À retenir : Architecture MVC, algorithmes mathématiques, UI responsive

---

🌐 TD 5 : API & Persistance (Java) (4h)

Objectifs pédagogiques

- Consommer des API REST avec Retrofit
- Gérer la persistance avec Room Database
- Implémenter le pattern Repository

Contenu théorique (1h)

5.1 Architecture de données

- **SQLite** : Base de données locale légère
- **Room** : ORM officiel Android, annotations @Entity, @Dao

- **Retrofit** : Client HTTP type-safe pour API REST
- **Pattern Repository** : Abstraction des sources de données

5.2 Gestion réseau mobile

- **Connectivité** : Détection réseau, gestion offline
- **Cache** : Stratégies de mise en cache
- **Sécurité** : HTTPS, authentification token

Travaux pratiques (3h)

Application Météo complète avec cache et synchronisation

Phase 1 : Configuration dépendances (20min)

gradle

```
// app/build.gradle
dependencies {
    implementation 'androidx.room:room-runtime:2.4.3'
    implementation 'androidx.room:room-rxjava3:2.4.3'
    annotationProcessor 'androidx.room:room-compiler:2.4.3'

    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'com.squareup.retrofit2:adapter-rxjava3:2.9.0'

    implementation 'io.reactivex.rxjava3:rxandroid:3.0.0'
    implementation 'io.reactivex.rxjava3:rxjava:3.0.1'

    implementation 'com.github.bumptech.glide:glide:4.14.2'
}
```

Phase 2 : Modèles de données (40min)

java

```java
// models/WeatherData.java
@Entity(tableName = "weather_data")
public class WeatherData {
    @PrimaryKey
    @NonNull
    public String city;

    public double temperature;
    public double feelsLike;
    public int humidity;
    public String description;
    public String icon;
    public double windSpeed;
    public int pressure;
    public double visibility;
    public long timestamp;

    // Constructeurs, getters, setters
    public WeatherData(@NonNull String city, double temperature, double feelsLike,
                       int humidity, String description, String icon, double windSpeed,
                       int pressure, double visibility) {
        this.city = city;
        this.temperature = temperature;
        this.feelsLike = feelsLike;
        this.humidity = humidity;
        this.description = description;
        this.icon = icon;
        this.windSpeed = windSpeed;
        this.pressure = pressure;
        this.visibility = visibility;
        this.timestamp = System.currentTimeMillis();
    }

    // Méthodes utilitaires
    public boolean isDataFresh() {
        long currentTime = System.currentTimeMillis();
        long dataAge = currentTime - timestamp;
        return dataAge < (30 * 60 * 1000); // 30 minutes
    }

    public String getFormattedTemperature() {
        return String.format("%.1f°C", temperature);
    }

    public String getIconUrl() {
        return "https://openweathermap.org/img/wn/" + icon + "@2x.png";
```

```
    }
}

// models/WeatherResponse.java (pour l'API)
public class WeatherResponse {
    public Main main;
    public Weather[] weather;
    public Wind wind;
    public String name;
    public long dt;

    public static class Main {
        public double temp;
        @SerializedName("feels_like")
        public double feelsLike;
        public int humidity;
        public int pressure;
        public double visibility;
    }

    public static class Weather {
        public String main;
        public String description;
        public String icon;
    }

    public static class Wind {
        public double speed;
        public int deg;
    }

    public WeatherData toWeatherData() {
        return new WeatherData(
            name,
            main.temp - 273.15, // Conversion Kelvin vers Celsius
            main.feelsLike - 273.15,
            main.humidity,
            weather[0].description,
            weather[0].icon,
            wind.speed,
            main.pressure,
            main.visibility / 1000.0 // Conversion en km
        );
    }
}
```

Phase 3 : API Service et Database (50min)

java

```java
// network/WeatherApiService.java
public interface WeatherApiService {
    String BASE_URL = "https://api.openweathermap.org/data/2.5/";
    String API_KEY = "votre_cle_api"; // À remplacer par votre clé

    @GET("weather")
    Observable<WeatherResponse> getCurrentWeather(
        @Query("q") String city,
        @Query("appid") String apiKey
    );

    @GET("forecast")
    Observable<ForecastResponse> getForecast(
        @Query("q") String city,
        @Query("appid") String apiKey,
        @Query("cnt") int count
    );
}

// database/WeatherDao.java
@Dao
public interface WeatherDao {
    @Query("SELECT * FROM weather_data WHERE city = :city LIMIT 1")
    Observable<WeatherData> getWeatherByCity(String city);

    @Query("SELECT * FROM weather_data ORDER BY timestamp DESC")
    Observable<List<WeatherData>> getAllWeatherData();

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insertWeather(WeatherData weatherData);

    @Delete
    void deleteWeather(WeatherData weatherData);

    @Query("DELETE FROM weather_data WHERE timestamp < :threshold")
    void deleteOldData(long threshold);

    @Query("SELECT COUNT(*) FROM weather_data")
    int getWeatherCount();
}

// database/WeatherDatabase.java
@Database(entities = {WeatherData.class}, version = 1, exportSchema = false)
@TypeConverters({Converters.class})
public abstract class WeatherDatabase extends RoomDatabase {
    private static volatile WeatherDatabase INSTANCE;
```

```
    public abstract WeatherDao weatherDao();

    public static WeatherDatabase getDatabase(final Context context) {
        if (INSTANCE == null) {
            synchronized (WeatherDatabase.class) {
                if (INSTANCE == null) {
                    INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
                            WeatherDatabase.class, "weather_database")
                            .build();
                }
            }
        }
        return INSTANCE;
    }
}
```

Phase 4 : Repository Pattern (60min)

java

```java
// repository/WeatherRepository.java
public class WeatherRepository {
    private WeatherApiService apiService;
    private WeatherDao weatherDao;
    private Context context;

    public WeatherRepository(Context context) {
        this.context = context;

        // Configuration Retrofit
        Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(WeatherApiService.BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .addCallAdapterFactory(RxJava3CallAdapterFactory.create())
            .build();

        apiService = retrofit.create(WeatherApiService.class);
        weatherDao = WeatherDatabase.getDatabase(context).weatherDao();
    }

    public Observable<WeatherData> getWeather(String city, boolean forceRefresh) {
        if (!forceRefresh) {
            // Tentative de récupération depuis le cache
            return weatherDao.getWeatherByCity(city)
                .filter(weather -> weather != null && weather.isDataFresh())
                .switchIfEmpty(fetchFromNetwork(city));
        } else {
            return fetchFromNetwork(city);
        }
    }

    private Observable<WeatherData> fetchFromNetwork(String city) {
        if (!isNetworkAvailable()) {
            return Observable.error(new Exception("Pas de connexion Internet"));
        }

        return apiService.getCurrentWeather(city, WeatherApiService.API_KEY)
            .map(WeatherResponse::toWeatherData)
            .doOnNext(weatherData -> {
                // Sauvegarder en cache
                new Thread(() -> weatherDao.insertWeather(weatherData)).start();
            })
            .onErrorResumeNext(error -> {
                // En cas d'erreur réseau, essayer de récupérer depuis le cache
                return weatherDao.getWeatherByCity(city)
                    .switchIfEmpty(Observable.error(error));
```

```
        });
    }

    public Observable<List<WeatherData>> getFavorites() {
        return weatherDao.getAllWeatherData();
    }

    public void addToFavorites(WeatherData weatherData) {
        new Thread(() -> weatherDao.insertWeather(weatherData)).start();
    }

    public void removeFromFavorites(WeatherData weatherData) {
        new Thread(() -> weatherDao.deleteWeather(weatherData)).start();
    }

    public void cleanOldData() {
        long threshold = System.currentTimeMillis() - (7 * 24 * 60 * 60 * 1000); // 7 jours
        new Thread(() -> weatherDao.deleteOldData(threshold)).start();
    }

    private boolean isNetworkAvailable() {
        ConnectivityManager connectivityManager =
            (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
        return activeNetworkInfo != null && activeNetworkInfo.isConnected();
    }
}
```

Phase 5 : Activity principale (50min)

java

```java
// MainActivity.java
public class MainActivity extends AppCompatActivity {

    private EditText cityInput;
    private Button searchButton;
    private TextView cityName, temperature, description, humidity, windSpeed;
    private ImageView weatherIcon;
    private RecyclerView favoritesRecycler;
    private ProgressBar loadingProgress;

    private WeatherRepository repository;
    private FavoritesAdapter adapter;
    private CompositeDisposable disposables = new CompositeDisposable();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        initializeViews();
        setupRepository();
        setupRecyclerView();
        setupClickListeners();
        loadFavorites();
    }

    private void initializeViews() {
        cityInput = findViewById(R.id.cityInput);
        searchButton = findViewById(R.id.searchButton);
        cityName = findViewById(R.id.cityName);
        temperature = findViewById(R.id.temperature);
        description = findViewById(R.id.description);
        humidity = findViewById(R.id.humidity);
        windSpeed = findViewById(R.id.windSpeed);
        weatherIcon = findViewById(R.id.weatherIcon);
        favoritesRecycler = findViewById(R.id.favoritesRecycler);
        loadingProgress = findViewById(R.id.loadingProgress);
    }

    private void setupRepository() {
        repository = new WeatherRepository(this);
    }

    private void setupRecyclerView() {
        adapter = new FavoritesAdapter(this::onFavoriteClick, this::onFavoriteDelete);
        favoritesRecycler.setLayoutManager(new LinearLayoutManager(this));
```

```
    favoritesRecycler.setAdapter(adapter);
}

private void setupClickListeners() {
    searchButton.setOnClickListener(v -> searchWeather());

    cityInput.setOnEditorActionListener((v, actionId, event) -> {
        if (actionId == EditorInfo.IME_ACTION_SEARCH) {
            searchWeather();
            return true;
        }
        return false;
    });
}

private void searchWeather() {
    String city = cityInput.getText().toString().trim();
    if (city.isEmpty()) {
        Toast.makeText(this, "Veuillez entrer une ville", Toast.LENGTH_SHORT).show();
        return;
    }

    showLoading(true);

    disposables.add(
        repository.getWeather(city, false)
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(
                this::displayWeather,
                this::handleError
            )
    );
}

private void displayWeather(WeatherData weather) {
    showLoading(false);

    cityName.setText(weather.city);
    temperature.setText(weather.getFormattedTemperature());
    description.setText(weather.description);
    humidity.setText("Humidité: " + weather.humidity + "%");
    windSpeed.setText("Vent: " + String.format("%.1f km/h", weather.windSpeed * 3.6));

    // Charger l'icône météo avec Glide
    Glide.with(this)
        .load(weather.getIconUrl())
```

```
            .placeholder(R.drawable.ic_weather_default)
            .into(weatherIcon);

        // Ajouter aux favoris automatiquement
        repository.addToFavorites(weather);
        loadFavorites();
    }

    private void handleError(Throwable error) {
        showLoading(false);
        Toast.makeText(this, "Erreur: " + error.getMessage(), Toast.LENGTH_LONG).show();
    }

    private void showLoading(boolean show) {
        loadingProgress.setVisibility(show ? View.VISIBLE : View.GONE);
        searchButton.setEnabled(!show);
    }

    private void loadFavorites() {
        disposables.add(
            repository.getFavorites()
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(
                    adapter::updateFavorites,
                    error -> Log.e("MainActivity", "Erreur chargement favoris", error)
                )
        );
    }

    private void onFavoriteClick(WeatherData weather) {
        displayWeather(weather);
        cityInput.setText(weather.city);
    }

    private void onFavoriteDelete(WeatherData weather) {
        repository.removeFromFavorites(weather);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        disposables.clear();
    }
}
```

À retenir : Architecture MVVM, RxJava, persistance et réseau

🐍 TD 6 : Android avec Python (Chaquopy) (3h)

Objectifs pédagogiques

- Intégrer Python dans une app Android avec Chaquopy
- Exploiter les bibliothèques Python pour le calcul scientifique
- Créer une interface hybride Java/Python

Contenu théorique (45min)

6.1 Chaquopy : Python sur Android

- **Installation** : Plugin Gradle, configuration build
- **Limitations** : Pas toutes les bibliothèques compatibles
- **Avantages** : Réutilisation code Python, calculs complexes

6.2 Communication Java ↔ Python

- **PyObject** : Interface Java pour objets Python
- **Sérialisation** : JSON pour structures complexes
- **Performance** : Considérations threading

Travaux pratiques (2h15)

Application de trading et analyse financière

Phase 1 : Configuration Chaquopy (30min)

```gradle
```

```
// app/build.gradle
plugins {
    id 'com.chaquo.python'
}

android {
    defaultConfig {
        python {
            buildPython "python3.9"
            pip {
                install "pandas"
                install "numpy"
                install "matplotlib"
                install "requests"
            }
        }
    }

    sourceSets {
        main {
            python.srcDirs = ["src/main/python"]
        }
    }
}
```

Phase 2 : Scripts Python (60min)

```
python
```

```python
# src/main/python/financial_analyzer.py
import json
import numpy as np
import pandas as pd
from datetime import datetime, timedelta
import requests

class FinancialAnalyzer:
    def __init__(self):
        self.api_key = "demo"  # Remplacer par vraie clé Alpha Vantage
        self.base_url = "https://www.alphavantage.co/query"

    def get_stock_data(self, symbol, period="1mo"):
        """Récupère les données d'une action"""
        try:
            # Simulation de données pour démo (remplacer par vraie API)
            dates = pd.date_range(end=datetime.now(), periods=30, freq='D')
            np.random.seed(42)  # Pour des résultats reproductibles

            # Générer des prix réalistes avec marche aléatoire
            initial_price = 100.0
            returns = np.random.normal(0.001, 0.02, len(dates))
            prices = [initial_price]

            for ret in returns[1:]:
                prices.append(prices[-1] * (1 + ret))

            data = {
                'symbol': symbol,
                'dates': [d.strftime('%Y-%m-%d') for d in dates],
                'prices': [round(p, 2) for p in prices],
                'volumes': [int(np.random.normal(1000000, 200000)) for _ in dates]
            }

            return json.dumps(data)

        except Exception as e:
            return json.dumps({"error": str(e)})

    def calculate_indicators(self, prices_json):
        """Calcule les indicateurs techniques"""
        try:
            data = json.loads(prices_json)
            prices = np.array(data['prices'])

            # Moyennes mobiles
```

```
        sma_20 = self.simple_moving_average(prices, 20)
        sma_50 = self.simple_moving_average(prices, 50)
        ema_12 = self.exponential_moving_average(prices, 12)

        # RSI
        rsi = self.calculate_rsi(prices)

        # MACD
        macd, signal = self.calculate
}

private void appendOperator(String operator) {
    String currentText = displayText.getText().toString();
    if (!currentText.isEmpty() && !isOperator(currentText.charAt(currentText.length() - 1))) {
        displayText.setText(currentText + " " + operator + " ");
        isNewCalculation = false;
    }
    resultText.setText("");
}

private void appendFunction(String function) {
    if (isNewCalculation) {
        displayText.setText(function);
        isNewCalculation = false;
    } else {
        String currentText = displayText.getText().toString();
        displayText.setText(currentText + function);
    }
}

private void appendDecimal() {
    String currentText = displayText.getText().toString();
    String[] parts = currentText.split(" ");
    String lastPart = parts[parts.length - 1];

    if (!lastPart.contains(".")) {
        if (isNewCalculation) {
            displayText.setText("0.");
            isNewCalculation = false;
        } else {
            displayText.setText(currentText + ".");
        }
    }
}

private void calculateResult() {
    try {
```

```
            String expression = displayText.getText().toString();
            double result = engine.evaluate(expression);

            // Formater le résultat
            String formattedResult;
            if (result == Math.floor(result)) {
                formattedResult = String.valueOf((long) result);
            } else {
                formattedResult = String.format("%.8f", result).replaceAll("0*$", "").replaceAll("\\.$", "");
            }

            resultText.setText("= " + formattedResult);

            // Ajouter à l'historique
            String historyEntry = expression + " = " + formattedResult;
            history.add(historyEntry);

            // Préparer pour le prochain calcul
            displayText.setText(formattedResult);
            isNewCalculation = true;

        } catch (Exception e) {
            resultText.setText("Erreur");
            Toast.makeText(this, "Expression invalide", Toast.LENGTH_SHORT).show();
        }
    }

    private void updateResult() {
        try {
            String expression = displayText.getText().toString();
            if (!expression.isEmpty() && !expression.endsWith(" ")) {
                double result = engine.evaluate(expression);
                String preview = String.format("%.4f", result);
                resultText.setText("≈ " + preview);
            }
        } catch (Exception e) {
            // Ignore les erreurs de preview
        }
    }

    private void clearAll() {
        displayText.setText("0");
        resultText.setText("");
        isNewCalculation = true;
    }

    private void deleteLastChar() {
```

```
String currentText = displayText.getText().toString();
if (currentText.length() > 1) {
    displayText.setText(currentText.substring(0, currentText.length() - 1));
} else {
    displayText.setText("0");
    isNewCalculation = true;
}
updateResult();
```