I.Les systèmes mobiles

Chaque téléphone intelligent, comme tout ordinateur, possède un **système d'exploitation** : c'est le logiciel qui est exécuté dès le démarrage de la machine et qui s'occupe de la gestion complète de celle-ci, notamment du lancement des autres logiciels (applications).

De même qu'il existe plusieurs systèmes d'exploitation pour ordinateurs personnels (Windows, macOS, Linux étant les plus connus), il existe plusieurs systèmes d'exploitation pour téléphones, la plupart étant également utilisés dans les tablettes tactiles, voire sur certains ordinateurs.

1. Android

Android est un système d'exploitation spécialisé au départ pour les téléphones mobiles. Datant de 2007 et développé par la société Google (à partir d'un projet développé par une start-up rachetée par Google), il est basé sur le noyau[1] Linux.

Android a fait le choix de la plateforme Java pour le développement, mais sans utiliser la machine virtuelle Java standard (propriété de la société Oracle). Jusqu'à la version 4.4 la machine virtuelle s'appelle *Dalvik*, puis ART (*Android RunTime*) à partir de la version 5.0. La bibliothèque standard d'Android ressemble beaucoup à J2SE (*Java 2 Standard Edition*) utilisé pour le développement « classique » sur ordinateurs, ce qui permet aux développeurs Java d'être familiarisés rapidement avec l'environnement.

D'abord destiné aux téléphones, le système Android est à présent disponible sur des tablettes, des ordinateurs de type PC, des montres, des téléviseurs, des box Internet…

C'est le système d'exploitation le plus utilisé dans le monde (en nombre d'appareils), c'est pourquoi il tiendra une place de choix dans cet ouvrage.

Les versions majeures d'Android se succèdent au rythme d'environ une par an.

---

[1] Le noyau d'un système d'exploitation est la partie chargée de la gestion du matériel, de la mémoire, de l'exécution des tâches.

2. iOS

iOS est le système d'exploitation, créé par Apple, qui équipe les appareils mobiles de la marque à la pomme : iPhone, iPad, iPod. C'est un système dérivé de macOS (le système d'exploitation des ordinateurs Apple), créé en 2007. La version actuelle est la 11, le système évoluant au rythme d'une version par an environ (les mises à jour de sécurité sont beaucoup plus fréquentes, évidemment). Il est obligatoire d'utiliser un ordinateur Apple sous macOS pour le développement d'applications pour iOS.

3. Windows 10

Windows est un système d'exploitation initialement destiné aux ordinateurs personnels de type PC, mais différentes branches de ce système ont été réalisées à destination des appareils mobiles (PDA puis téléphones).

Depuis la version 10 de Windows, son éditeur Microsoft a unifié ses différentes branches : c'est la plateforme Windows universelle (UWP : *Universal Windows Platform*) qui permet d'exécuter avec le même code, des applications sur PC, téléphone mobile, console de jeu Xbox, etc.

La présence de Windows sur des téléphones, bien qu'ancienne, n'a jamais été à la hauteur des deux autres plateformes (Android et iOS) et est en chute libre ces dernières années. Néanmoins, le côté universel de la plateforme la rend intéressante : développer pour téléphone n'est ici plus spécifique ! C'est pourquoi il tiendra une place dans cet ouvrage malgré des parts de marché faibles sur téléphone (mais meilleures sur tablettes, et très bonnes sur ordinateurs).

Il est obligatoire d'utiliser un ordinateur sous Windows 10 pour pouvoir développer pour un appareil Windows 10.

4. Ancêtres et exotiques

Malgré ce que pense souvent le grand public, les téléphones intelligents ne sont pas apparus avec l'iPhone ! Avant même les premiers téléphones programmables, il existait des appareils qui, mis à part la fonction de téléphonie, en étaient proches : les PDA (*personal digital assistant*).

Un des premiers systèmes d'exploitation pour appareils mobiles était *Palm OS*. Créé en 1995, ce système d'exploitation embarqué était utilisé

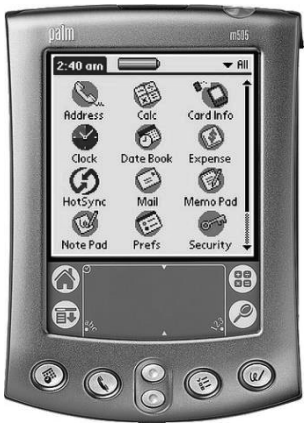

Image 1 : PDA sous PalmOS

dans des PDA, des smartphones, des montres, des GPS de voiture, des lecteurs de code-barres, des consoles de jeux. Il utilisait un écran tactile utilisable au doigt ou à l'aide d'un stylet. La programmation se faisait en général en C ou C++, mais il existait une machine virtuelle Java pour *Palm OS* (donc la possibilité de faire des applications en Java), des compilateurs Pascal et Basic et quelques autres langages plus exotiques. *Palm OS* est ensuite devenu *Palm WebOS*, puis *HP WebOS* et les derniers appareils sous ce système sont sortis en 2011.

Microsoft, leader dans les systèmes d'exploitation pour ordinateurs, a sorti en 1996 un système d'exploitation spécifique pour appareils mobiles appelé *Windows CE*. Utilisé dans des PDA, des téléphones, des DAB[1], des terminaux de paiement, des lecteurs de code-barres, des consoles de jeu, des GPS de voiture, le système a été très utilisé dans les premières générations d'appareils mobiles. Le système a ensuite évolué en *Windows mobile* mais a été abandonné en 2012 pour être remplacé par *Windows phone* (basé lui sur un autre noyau, donc totalement différent malgré le nom proche), lui-même remplacé par *Windows 10 universel*.

Image 2 : PDA sous Windows CE

La fondation Mozilla, éditrice du navigateur *Firefox*, a conçu un système d'exploitation pour appareils mobiles appelé *Firefox OS*. Il est basé, comme Android, sur le noyau Linux et se comporte en gros comme un navigateur : les applications doivent donc être écrites en HTML5. La première version date de 2013. Très peu de smartphones ont été vendus avec *Firefox OS*, mais le projet est toujours actif.

[1] Distributeur Automatique de Billets.

Il existe d'autres systèmes pour mobile, en général dérivés d'autres systèmes d'exploitation ou de navigateurs : *Ubuntu Touch*, *Sailfish OS*, *Tizen*, *Chromium OS*, *Google Chrome OS*…

II. Rappels de programmation objet

Le paradigme de programmation objet étant particulièrement employé dans la programmation sur terminaux mobiles (téléphones, tablettes…), comme dans d'autres domaines, quelques rappels me semblent importants. Si bien entendu vous êtes à l'aise avec ces notions, n'hésitez pas à passer directement au chapitre suivant !

1. Notion d'objet, de classe

Un **objet** est une représentation d'une notion du monde réel. C'est une variable, au sens programmation du terme, occupant une certaine zone de la mémoire centrale de la machine. Le paradigme objet revient, en gros, à considérer un programme comme un ensemble d'objets qui évoluent, communiquent et collaborent pour effectuer le travail voulu.

Un objet possède un certain **état** qui, au long du déroulement du programme, change. L'ensemble des états de tous les objets du programme représente l'état global du programme. Un programme peut alors être vu comme une machine à états, un automate.

Un objet possède également un certain **comportement**, c'est-à-dire un ensemble d'opérations qui permettent de modifier son état ou de collaborer avec l'objet.

Une **classe** est en quelque sorte le regroupement des objets de même nature (même type d'état, même comportement). C'est un **type** qui caractérise l'objet.

Une classe possède donc :

- Des **attributs** qui représentent son état : ce sont des variables
- Des **opérations** qui représentent son comportement ; on peut les regrouper en différents types :
    - Les **constructeurs** définissent l'état initial de l'objet
    - Les **mutateurs** modifient l'état de l'objet
    - Les **accesseurs** permettent de connaitre l'état d'un objet
    - Les **destructeurs** mettent fin à la vie de l'objet

Chaque membre d'une classe possède un certain niveau d'accès. Les niveaux courants sont les suivants :

- **Public** : tout objet peut accéder au membre
- **Protégé** : seuls les objets du type ou d'un type hérité peuvent accéder au membre
- **Privé** : seuls les objets de la même classe peuvent accéder au membre
- **Paquet** : seuls les objets du même paquet peuvent accéder au membre

Les accès courants sont :

- Privé pour les attributs et les opérations d'implémentation (détails du code)
- Public pour les opérations de l'interface
- Protégé pour les opérations d'implémentation abstraites ou les propriétés internes.

a) Représentation UML

Le langage UML est un langage de représentation graphique. Il est particulièrement adapté au paradigme objet. Une classe en UML se représente par un rectangle comportant trois compartiments :

- Le nom de la classe (avec un éventuel stéréotype)
- Les attributs de la classe
- Les opérations de la classe

Un attribut possède un certain type (entier, caractère, etc.), peut avoir une valeur initiale ainsi que certaines caractéristiques (constant, unique, etc.).

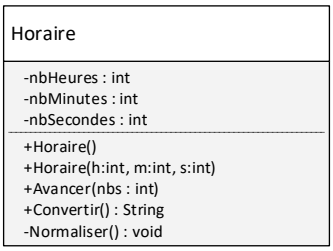

Diagramme 1 : une classe en UML

Une opération, parfois appelée méthode ou fonction, possède un certain type de retour (`void` si aucun), des paramètres et certaines caractéristiques.

La classe ci-contre (`Horaire`) représente un horaire dans la journée. Elle possède en attributs trois entiers qui permettent de stocker son état (nombre d'heures, de minutes, de secondes). Son

comportement comprend deux constructeurs (l'un sans paramètre et l'autre avec trois paramètres entiers), un accesseur (pour obtenir l'état sous la forme de texte) et deux mutateurs (un pour faire avancer l'horaire, et l'autre pour le normaliser) dont l'un, privé, est réservé à l'usage interne de la classe.

b) Une classe en Java

Le langage Java, utilisé notamment pour la programmation native Android (voir page 49), suit le paradigme objet et peut donc représenter une classe.

L'extrait de code suivant correspond au Diagramme 1 :


```
class Horaire {
     private int nbHeures;
     private int nbMinutes;
     private int nbSecondes;

     public Horaire()
     {
        nbHeures = nbMinutes = nbSecondes = 0;
     }
     public Horaire(int h, int m, int s)
     {
        nbHeures = h;
        nbMinutes = m;
        nbSecondes = s;
        normaliser();
     }
     public void avancer(int nbs)
     {
        nbSecondes += nbs;
        normaliser();
     }
     public String convertir()      {
        return String.valueOf(nbHeures)+
          ":"+String.valueOf(nbMinutes)+
          ":"+String.valueOf(nbSecondes)
     }
     private void normaliser(){
        nbMinutes += nbSecondes/60;
        nbSecondes %= 60;
        nbHeures += nbMinutes/60;
        nbMinutes %=60;
        nbHeures %= 24;
     }
}
```

c) Une classe en Swift

Swift est le langage natif pour développer sous iOS. En dehors de l'univers Apple, c'est un langage très peu utilisé et il n'est donc pas utile d'apprendre ce langage si vous ne développez pas pour les appareils à la pomme.

L'extrait de code suivant correspond au Diagramme 1 :


```
public class Horaire {
    private var nbHeures : Int
    private var nbMinutes : Int
    private var nbSecondes : Int

    public init()     {
        nbHeures = 0
        nbMinutes = 0
        nbSecondes = 0
    }

    public init(h:Int, m:Int, s:Int)
    {
        nbHeures = h
        nbMinutes = m
        nbSecondes = s
    }

    public func avancer(nbs:Int)
    {
        nbSecondes += nbs
        normaliser()
    }

    public func convertir() -> String
    {
        return String(nbHeures) + ":"  + String(nbMinutes) +
     ":" + String(nbSecondes)
    }

    private func normaliser()    {
        nbMinutes += nbSecondes/60
        nbSecondes %= 60
        nbHeures += nbMinutes/60
        nbMinutes %= 60
        nbHeures %= 24
    }
}
```