Développement mobile MIDP 2.0

Frédéric BERTIN
fbertin@neotilus.com

Intro : panorama des plateformes mobiles

Panorama des plateformes mobiles

Plateforme Langages	Positif	Négatif
Windows Mobile dotNET – C++	•Performance (code natif c++)	•Standard fermé.
Symbian OS S60 C++	•Performance (code natif c++)	•Très peu portable, même sur des versions différentes Symbian. •(A suivre PIPS=POSIX)
MIDP Java	•Standard ouvert (Nokia, Sony-Ericsson, Samsung, Motorola, …). •Le plus grand parc de téléphones	•Non homogène •De + en + de versions 1.0, 2.0, 2.1, 3.0, … •Librairies supplémentaires.
DoJa Java	•Homogène •Spec Hardware	•Standard fermé (DoCoMo, iMode alliance) •très peu d'opérateurs (Bouygues)
Androïd **(Open Handset Alliance)** C++ - Java	•Performance (code natif c++) •Open Source (Licence Apache v2)	•1 device : HTC G1
iPhone C – Objective C – (Java ?)	•Homogène •Performances	•2 devices : iPhone et iPhone 3G

Plateformes : les outils de développement

Plateforme	Environnement de developpement	SDK
Windows Mobile	Visual Studio	Pocket PC SDK (WM5), Smartphone SDK (WM5), Windows Mobile 6 SDK : http://msdn.microsoft.com/windowsmobile
Symbian OS S60	CarbideC++ (basé sur Eclipse)	Symbian SDK http://forum.nokia.com
MIDP	Eclipse + MTJ	Sun WTK : http://java.sun.com/products/sjwtoolkit/
DoJa	Eclipse	iAppli Tools : http://www.doja-developer.net/
Androïd (Open Handset Alliance)	Eclipse	Androïd SDK : http://code.google.com/intl/fr/android/download.html
iPhone	Xcode	iPhone SDK : http://developer.apple.com/iphone/

Présentation J2ME / MIDP

Midp 1.0 : Limitations

- Pas de support de virgule flottante (FPU) : CLDC 1.0
- Pas de support audio (beeps, et c'est tout)
- Graphiques : pas de support de la transparence
- Graphiques : faiblesse du design GUI pour les jeux
- Réseau : ne supporte que le HTTP

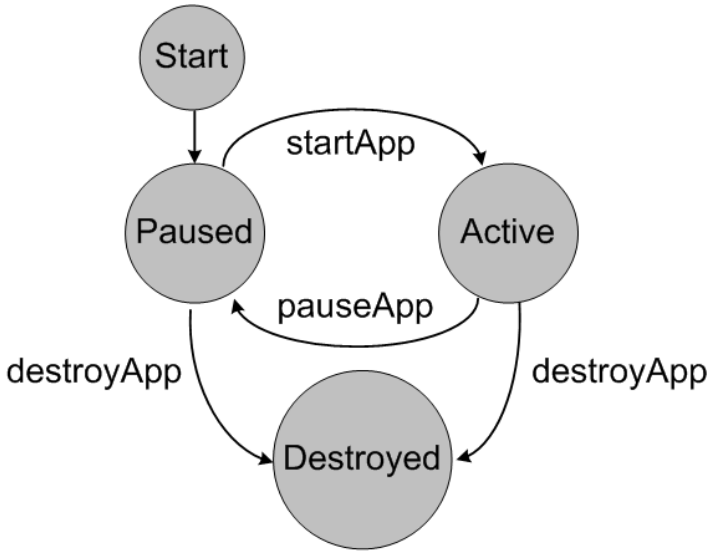
Midp 2.0 : Nouveautés

- Support multimédia (MMAPI)
- Package dédié aux jeux (game API)
- Réseau : HTTP, HTTPS, socket, port série, ….
- Architecture Push (PUSH registry)
- OTA Provisioning
- Sécurité

J2ME / MIDP : les API

MIDP 2.0 : Les MIDlets

- Package `javax.microedition`
- La classe de base de toute application MIDP.
- Toutes les applications MIDP héritent de la classe abstraite `javax.microedition.MIDlet`
- Trois méthodes sont appelées pour prendre en charge le cycle de vie de la MIDlet
  - startApp()
  - pauseApp()
  - destroyApp()

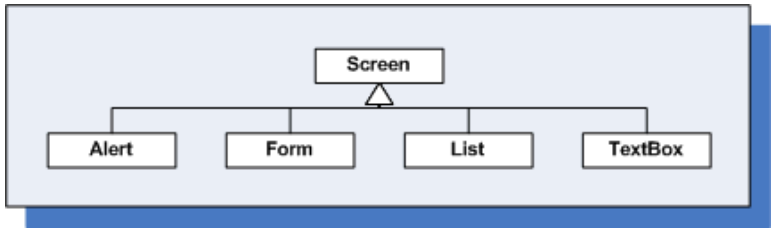


- Similitude avec le cycle de vie d'une applet Java

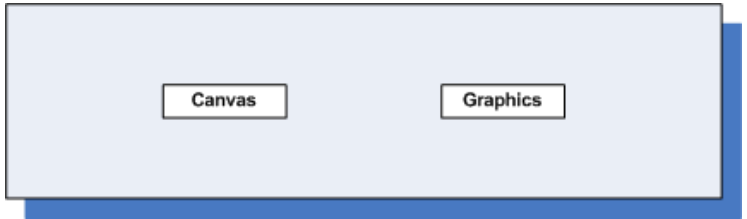
MIDP 2.0 IHM : Les composants graphiques

- Package `javax.microedition.lcdui`

- API haut niveau
  - Formulaires
  - « Look and feel » minimal, pris en charge par la JVM
  - Indépendant du téléphone



- API bas niveau
  - Contrôle précis de l'affichage et de la position
  - Très dépendant du téléphone

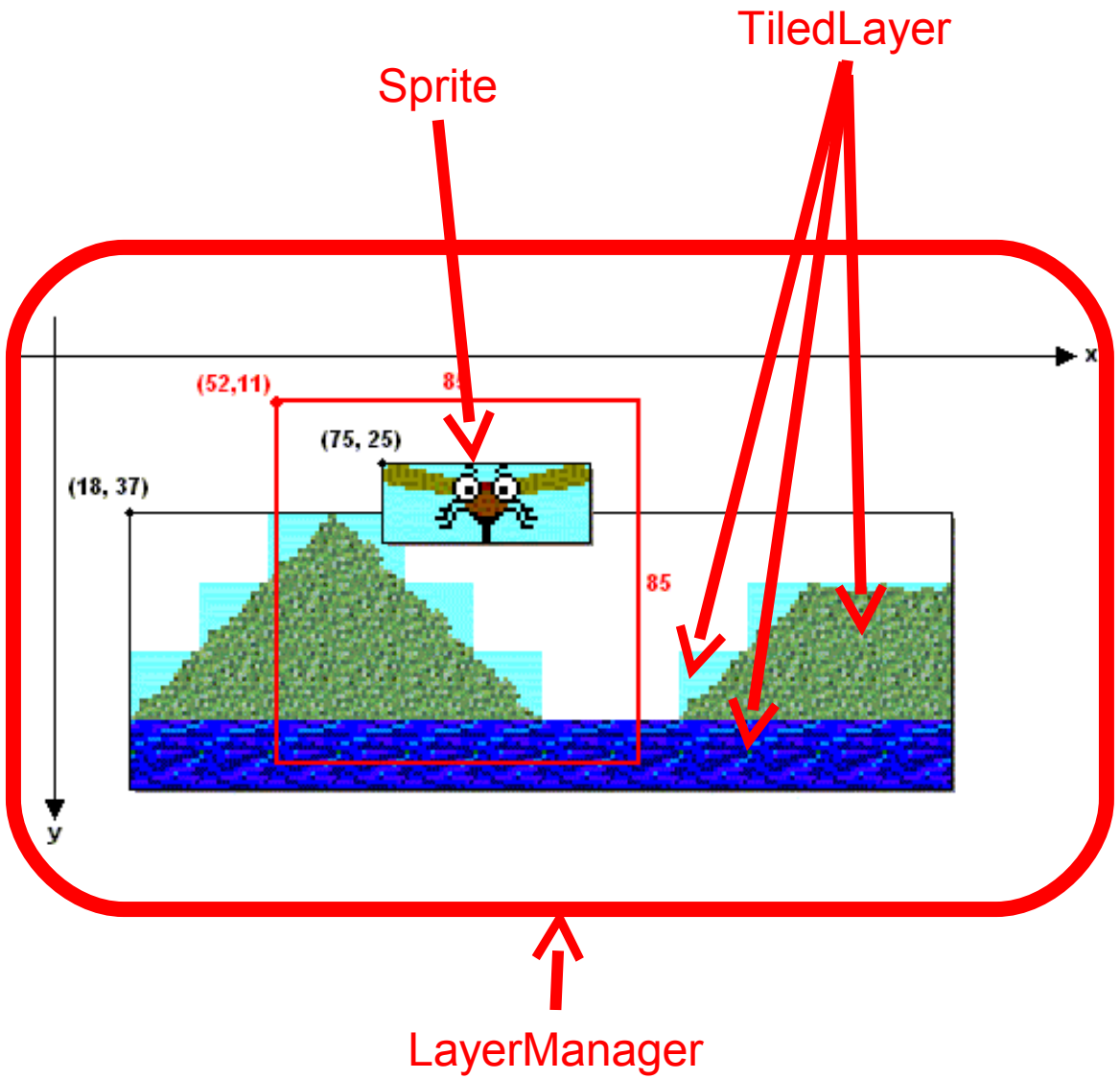


MIDP 2.0 Game API

- API dédiée aux jeux
  - Augmentation des performances
  - Réduction de code : prise en charge de la logique du jeu
  - Basés sur des éléments graphiques bas niveau (`Graphics, Image,Canvas`).

- Package `javax.microedition.lcdui.game`, 5 classes à connaitre par cœur ☺:
  - GameCanvas
    - Gère l'affichage du jeu, sa logique
  - Layer
    - Couche graphique (transparence)
  - LayerManager
    - Gère une série de layers
  - Sprite
    - Gère les animations
  - TiledLayer
    - Utile pour les maps

MIDP 2.0 Game API

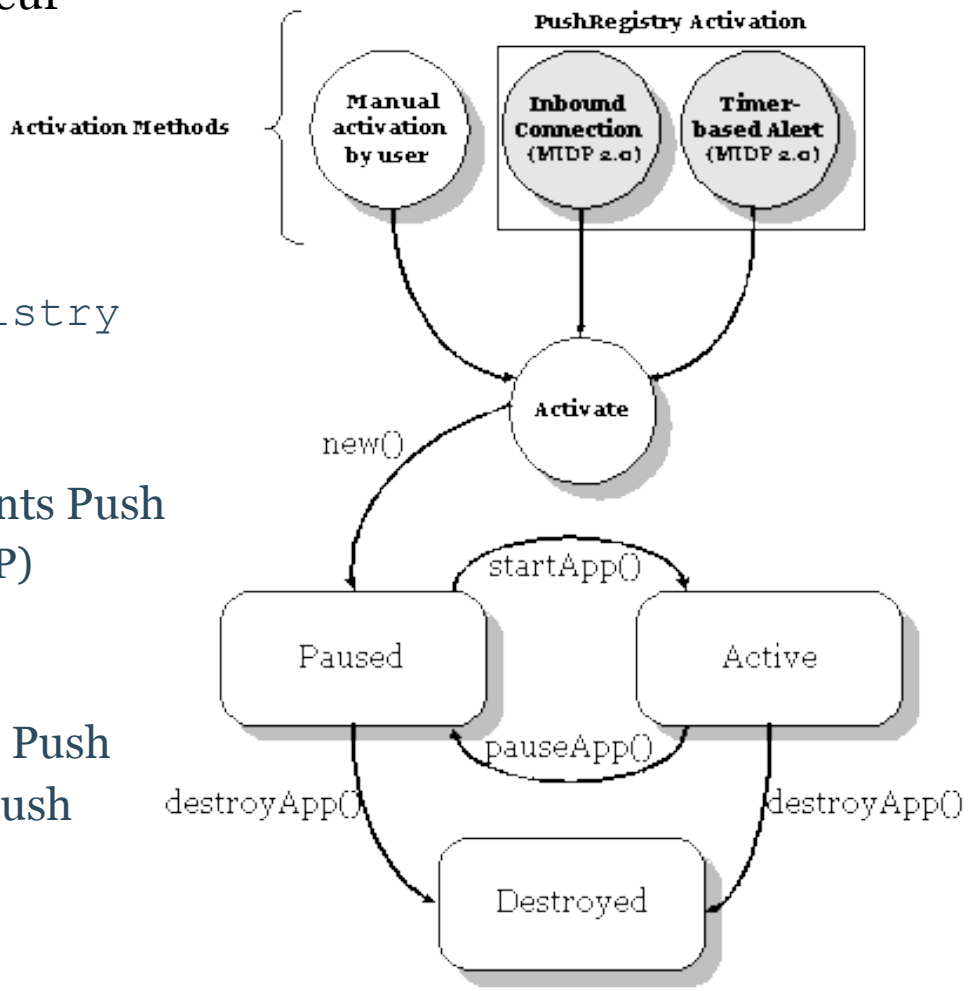


MIDP 2.0 : Push Registry

- Lance une midlet sans interaction utilisateur
  - Activation par réseau
  - Activation par timer

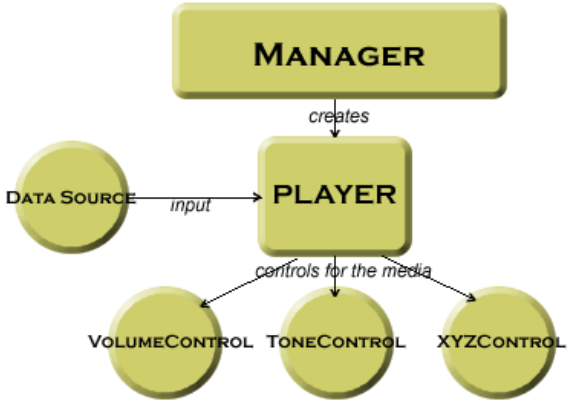
- Une seule classe :
  - `Javax.microedition.io.PushRegistry`

- Avec l'API PushRegistry on peut :
  - Enregistrer une midlet sur des événements Push
    - Connection entrante (Socket TCP, UDP)
    - SMS
    - Timer
  - Découvrir si une midlet a été activée par Push
  - Retrouver des paramètres d'activation Push



MMAPI (JSR 135)

- Permet l'enregistrement et la lecture de fichiers multimedias
  - Lecture de fichiers packagés avec la midlet (wave, midi, videos, …)
  - Lecture de fichiers internes au mobile
  - Lecture de fichiers accessible via le réseau (streaming)
  - Prise de photos, …



- Package `javax.microedition.media`, les classes principales :
  - Player
  - Control
  - Manager
  - PlayerListener
- Package `javax.microedition.media.control`
  - VolumControl
  - ToneControl
  - VideoControl
  - …

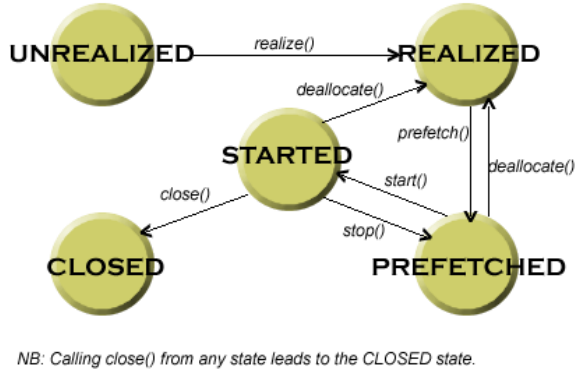
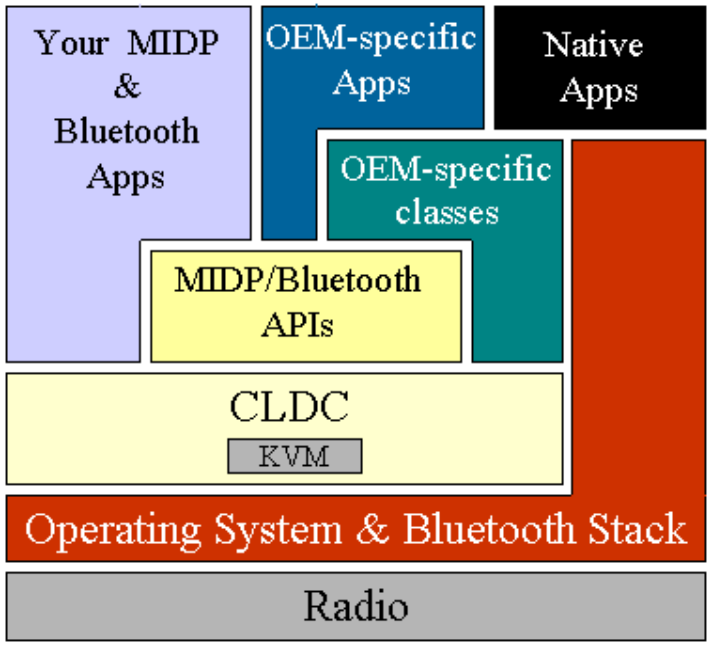


Diagramme Etat / transition du player

Bluetooth (JSR 82)

- API d'accès aux périphériques bluetooth
  - Specs bluetooth 1.1
  - Découvrir des périphériques BT
  - S'enregistrer auprès de peripheriques BT
  - Etablir une connexion BT, et échanger des données
- Package `javax.bluetooth`
  - Core bluetooth API
- Package `javax.obex`
  - Object Exchange (pas spécifique bluetooth)
- Programmation d'une application bluetooth
  1. Initialisation stack BT
  2. Device management
  3. Device discovery
  4. Service discovery
  5. Communication (OBEX, RFCOMM, L2CAP)

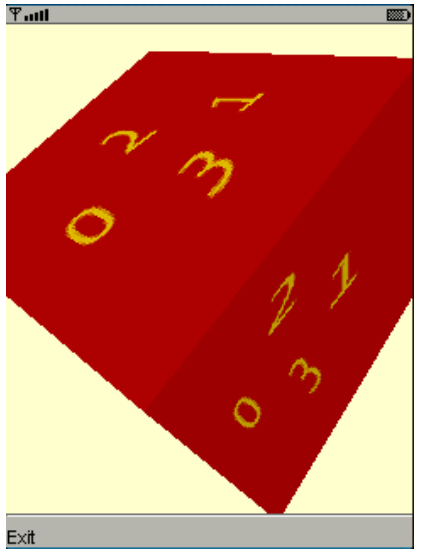


Localisation (JSR 179)

- Offre des services basés sur la localisation du téléphone
  - Ou je suis ?
  - Qu'y a-t-il autour de moi ?
  - …
- Nécessite CLDC 1.1 (FPU)
- Utilise plusieurs techniques pour localiser le téléphone
  - GPS
  - Cell ID (réseau mobile)
  - Bluetooth
- Package `javax.microedition.location`
  - Enregistre la positions
  - Enregistre la direction(boussoles)
  - Peut sauvegarder, et enregistrer des POI sur le téléphone

Mobile 3D Game API (JSR 135)

- Requiert CLDC 1.1 (pour le FPU)
- Fournit une interface de bas et haut niveau 3D
  - Création de jeux,
  - Création d'IHM,
  - Messagerie, avatars, …
- Package `javax.microedition.m3g`



J2ME / MIDP :
Le développement

Déploiement d'une midlet suite

- Le **.jad** (fichier de description) est téléchargé, précisant
  - L'URL de téléchargement du .jar
  - La taille de l'application
  - Le nom du jeu
  - La midlet de démarrage (qui étend MIDLet)
  - La version
  - Les resssouces, les protocoles utilisés, …

- Le **.jar** est téléchargé
  - Un seul .jar
  - Ne peut utiliser que les bibliothèque présentes dans le téléphone
  - Pas de partage de ressources entre MIDlets téléchargées

3. Si l'application a besoin d'accéder à des ressources (réseau), prompt utilisateur

5. L'application se lance

MIDP 2.0 : Sécurité

- Packages nécessitant des autorisations spéciales :

```
javax.microedition.io.Connector.http
javax.microedition.io.Connector.socket
javax.microedition.io.Connector.https
javax.microedition.io.Connector.ssl
javax.microedition.io.Connector.datagram
javax.microedition.io.Connector.serversocket
javax.microedition.io.Connector.datagramreceiver
javax.microedition.io.Connector.comm
javax.microedition.io.PushRegistry
```

- Nécessite de spécifier les permissions de la midlet dans le .jad, exemple :

```
MIDlet-Permissions: javax.microedition.io.Connector.http
MIDlet-Permissions-opt: javax.microedition.io.Connector.socket
```

- Définissions de domaines « trusted » et « untrusted »
  - Signature du code souvent nécessaire
  - Basé sur la notion de certificats

Java Mobile, les pièges

- « Write Once, Run Everywhere » : A OUBLIER !! :o)
- Les JVM sont souvent exotiques
- Les JSR sont souvent implémentées partiellement
- S'ecrire un logger sur le téléphone peut aider à débugger.
- Le comportement de l'application sur l'emulateur peut être différent sur le device
- Tester, tester, re-tester (sur tous les téléphones cibles)

Tips J2ME : Attention à la mémoire

- Limiter le nombre de classes (pas plus de 10 !), les regrouper.
- Sortir les constantes du code dans un fichier de conf (binaire), éviter les `static final`
- Bannir les `string` au maximum, elles apparaissent en clair dans les .class, donc augmentent leur taille.
- Obfusquer le code (le compresser)
  - Réécritures des noms de classe, des méthodes, des champs
    - Ex : `MaFonction()` devient `1()`
  - Mais garde les noms des packages !
  - Exemples d'outils
    - ProGuard (outil open source)
    - RetroGuard
    - ...

Tips J2ME : ménager le processeur

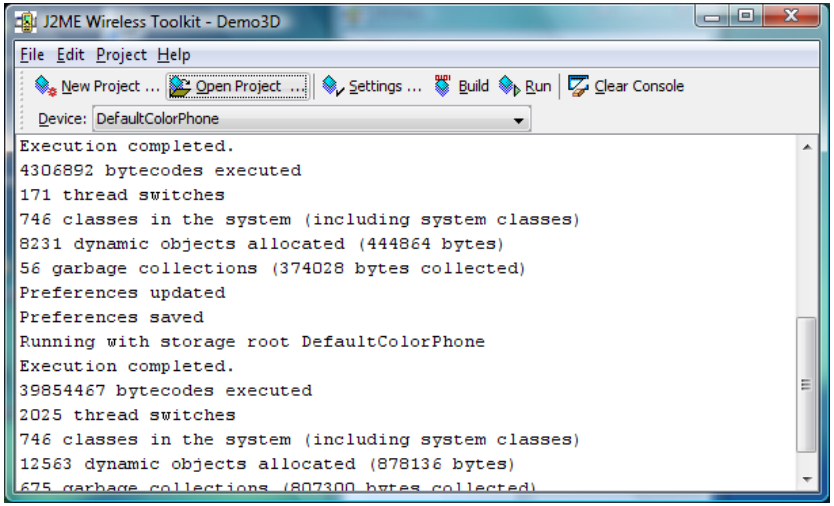
- Regrouper les créations d'objets (« `new` ») à l'initialisation de l'application.
- Recycler les objets :
  - On créé au démarrage de l'application un pool d'Objets, et on les cast
  - Travail à mémoire constante
- Eviter au maximum le déclenchement du `garbage collector`.
  - Le `garbage collector` est couteux !
  - Un lancement du `garbage collector` peut freezer l'application !!

Développement : Sun WTK

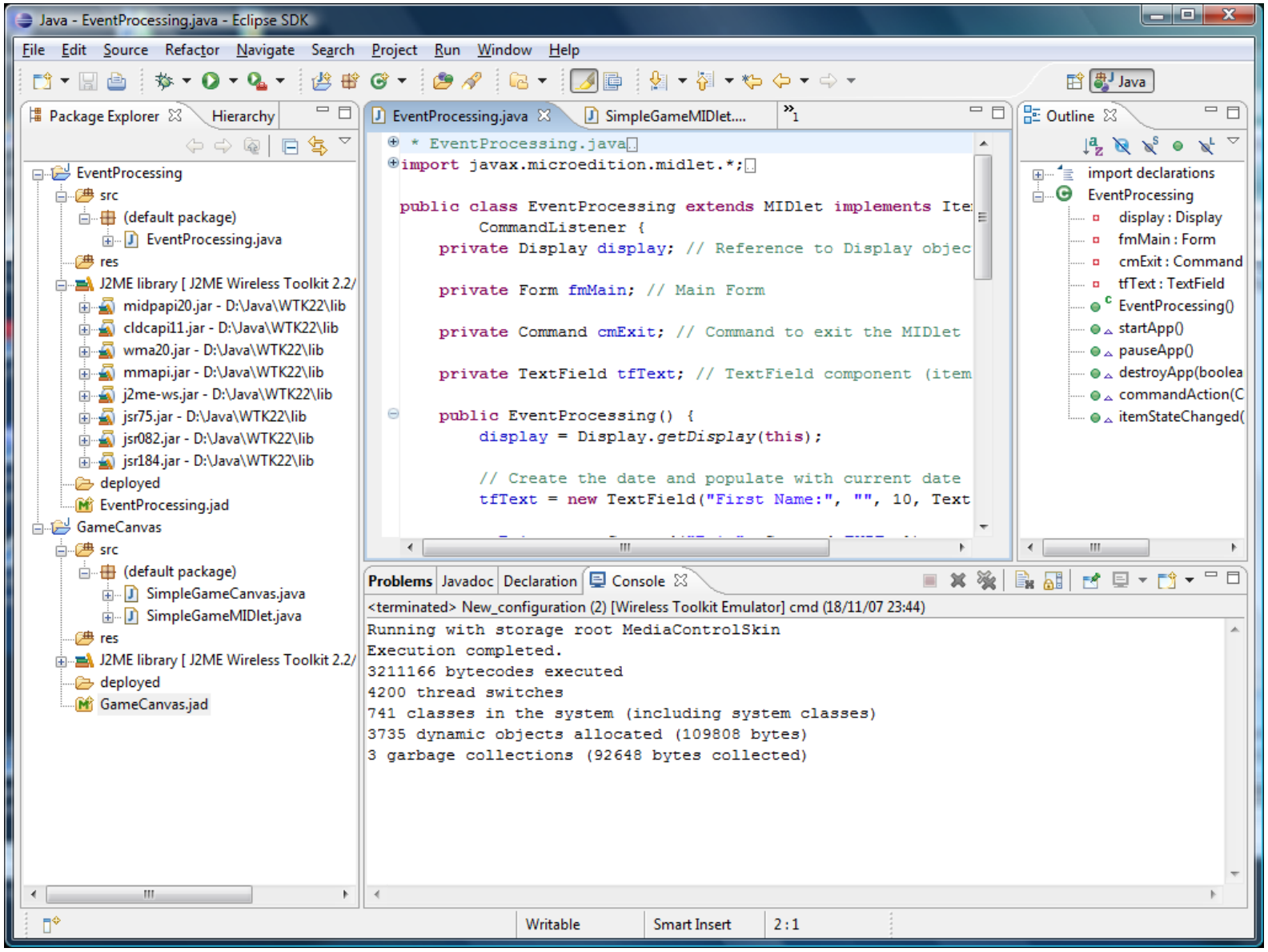
- Environnement de développement minimal :kToolbar
  - Compilation, packaging, exécution dans un emulateur
- Suffisant pour des tests
- 1 projet kToolbar = 1 midlet suite
- Possibilités de profiling (mémoire, appels système)

Inconvénients

- N'a pas de possibilités de debug fin (breakpoints)
- Possibilités d'édition minimales (type notepad)



Développement : Eclipse + MTJ



Questions / Réponses

TP MIDP 2 : API graphique« Haut niveau »

- Créer une Midlet
- Créer un formulaire
  - Bouton Exit (Command, CommandListener)
  - Saisie password (TextField)
  - Affichage String (StringItem)
- Au changement du password, l'afficher en clair (interception d'événements sur le champ Password)

TP MIDP 2 : API graphique« Bas niveau »

- Créer une Midlet
- Créer un canvas
- Dans le paint() du canvas
  - Remplir l'écran d'une couleur
  - Tracer un carré au milieu de l'écran
  - Afficher une string au dessus du carré