IFT 1015 - API

Professeur:

Stefan Monnier

B. Kégl, S. Roy, F. Duranleau, S. Monnier

Département d'informatique et de recherche opérationnelle

Université de Montréal

hiver 2006

Au programme

Définition d'*API*

Utilité

API d'un langage

API externe

Documentation

Utilisation

Compromis

Définition

API*: *Application Programming Interface

(interface de programmation d'application)

Un *API* est une collection de fonctions et/ou de classes qui peuvent être utilisées par un programme.

En plus du code, cela inclu normalement:

- une documentation des fonctions publiques.
- une garantie parfois implicite que le comportement de ces fonctions restera stable.

Utilité des APIs

Pourquoi les *API*s existent? Pourquoi en faire usage?

Utilité des APIs

Pourquoi les *API*s existent? Pourquoi en faire usage?

$\longrightarrow$ Pour ne pas avoir à retaper constamment du code pour des tâches fréquentes.

$\Rightarrow$ **Réutilisation** !

$\longrightarrow$ Permet au programmeur de se concentrer sur des aspects plus important de ses programmes.

$\Rightarrow$ Rapidité de developpement (quoique...).

API d'un langage

Tout langage de programmation possède un *API* de base comprenant au moins des fonctions pour:

- les entrées et sorties;
- la manipulation de chaînes de caractères;
- les fonctions mathématiques fréquentes;
- communiquer avec le système d'exploitation;

i.e. l'essentiel pour *survivre*.

L'*API* souvent inclu aussi structures de données, *UI*, concurrence, ...

Java est assez légendaire pour son *API* monstre.

API externe

L'*API* du langage ne couvre pas tout.

$\Rightarrow$ On développe des bibliothèques (*libraries*) de fonctions ou classes spécialisées:

- graphisme 3D
- interface usager (*GUI*, *TUI*)
- traitement d'images
- traitement de la langue naturelle
- ...

Ces bibliothèques sont développées pour des langages précis, bien qu'il soit généralement possible de les utiliser avec d'autres langages.

Chercher la documentation

Livres sur un langage

Internet (site web de documentation, moteur de recherche, . . .)

Sous Unix: les commandes **`man`** et **`info`**!

Pour Java, le site de référence est java.sun.com

Consultez toujours la documentation!!!

Usage d'un API

- Pour faire usage d'un *API*, il faut indiquer au programme les fonctions ou classes qu'on désire utiliser.
- La procédure exacte varie d'un langage à l'autre.
- Pour plusieurs langages, on n'a pas besoin de faire quoique ce soit de particulier pour faire usage de l'*API* de base.

Ex.: Java (sauf pour les entrées/sorties).

Usage de l'API de Java

En Java, l'*API* est réparti dans des *packages*. Pour faire usage d'une classe dans un *package*, on écrit son nom devant le nom de la classe, ou bien on l'*importe*.

Pour *importer* une classe, écrire *avant* la déclaration d'une classe:

```
import package.Classe;
```

Pour importer toutes les classes d'un *package*, on écrit:

```
import package.*;
```

Tout ceci est facultatif pour les classes dans `java.lang`.

Exemple: Trier des nombres

Avec le `Arrays.sort(int[])` du *package* `java.util`:

```
public class TriArg {
  public static void main (String[] arg)
  {
    int[] tab = new int[arg.length];
    for (int i = 0; i < tab.length; ++i)
      tab[i] = Integer.parseInt(arg[i]); // Conversion en in

    java.util.Arrays.sort(tab);          // Effectue le tri.

    for (int i = 0; i < tab.length; ++i)
      System.out.print(tab[i] + " ");    // Affichage trié.
    System.out.println();
} }
```

Exemple: avec `import`

```
import java.util.Arrays;

public class TriArg {
  public static void main(String[] arg)
  {
    int[] tab = new int[arg.length];
    for (int i = 0; i < tab.length; ++i)
      tab[i] = Integer.parseInt(arg[i]); // Conversion en in

    Arrays.sort(tab);                    // Effectue le tri.

    for (int i = 0; i < tab.length; ++i)
      System.out.print(tab[i] + " ");    // Affichage trié.
    System.out.println();
} }
```

Exemple: en C++

```
#include <algorithm>
#include <cstdlib>
#include <iostream>
using namespace std;
int main (int argc, char* argv[])
{
  int tab[argc - 1];
  for (int i = 1; i < argc; ++i)
    tab[i - 1] = atoi(argv[i]);       // Conversion en int.
  sort(tab, tab + argc - 1);          // Effectue le tri.
  for (int i = 0; i < argc - 1; ++i)
    cout << tab[i] << " ";            // Affichage trié.
  cout << endl;
  return 0;
}
```

Compromis

Temps de recherche *vs* temps de codage

Il arrive que de coder ce qu'on veut soit plus rapide que de chercher la fonction qui le ferait dans l'*API*.

Compromis

Temps de recherche *vs* temps de codage

Il arrive que de coder ce qu'on veut soit plus rapide que de chercher la fonction qui le ferait dans l'*API*.

Temps de modification/adaptation de l'*API*

L'*API* ne correspond pas nécessairement à ce qu'on veut

$\Rightarrow$ l'adaptation et/ou la modification peut prendre plus de temps que de le faire soi-même.

Compromis

Temps de recherche *vs* temps de codage

Il arrive que de coder ce qu'on veut soit plus rapide que de chercher la fonction qui le ferait dans l'*API*.

Temps de modification/adaptation de l'*API*

L'*API* ne correspond pas nécessairement à ce qu'on veut

$\Rightarrow$ l'adaptation et/ou la modification peut prendre plus de temps que de le faire soi-même.

Temps de compréhension

Pour bien utiliser un *API*, il faut bien comprendre le comportement de ses fonctions et classes, sinon on peut se retrouver avec un programme qui ne fait pas ce qu'on veut ou qui est beaucoup plus lent que prévu.

Compromis

Temps de recherche *vs* temps de codage

Il arrive que de coder ce qu'on veut soit plus rapide que de chercher la fonction qui le ferait dans l'*API*.

Temps de modification/adaptation de l'*API*

L'*API* ne correspond pas nécessairement à ce qu'on veut

$\Rightarrow$ l'adaptation et/ou la modification peut prendre plus de temps que de le faire soi-même.

Temps de compréhension

Pour bien utiliser un *API*, il faut bien comprendre le comportement de ses fonctions et classes, sinon on peut se retrouver avec un programme qui ne fait pas ce qu'on veut ou qui est beaucoup plus lent que prévu.