Encodage et Fichier

Victor Marsault
Aldric Degorre

CPOO 2015

Outline

Qu'est-ce qu'une chaîne de caractère ?

En C

- tableau de `char` (donc de type `char*`) ;
- la longueur n'est pas indiquée, la chaîne s'arrête quand on rencontre le caractère '0'.
- un `char` est une suite de 8 bits : un entier entre 0 et 255.

En Java

- L'équivalent du `char` de C est `byte` (traduit octet en français) (le `char` de java est codé sur 16 bits)
- `String` est une chaîne sémantique :
  - Sa longueur est le nombre de caractère qu'il est censé s'afficher
  - (Son encodage est fixé à UTF-16.)
  - immutable (on ne peut modifier un certain caractère)
    $\rightarrow$ Partage d'espace mémoire.

Encodage

Un encodage est une manière d'**interpréter** un tableau d'octets.

Exemples :

- ASCII : 8ème bit est inutile, aucun caractère accentués.

Encodage

Un encodage est une manière d'**interpréter** un tableau d'octets.

Exemples :

- ASCII : 8ème bit est inutile, aucun caractère accentués.
- Extended ASCII : ajouts de quelques caractères accentués et de caractères d'affichages.

Encodage

Un encodage est une manière d'**interpréter** un tableau d'octets.

Exemples :

- ASCII : 8ème bit est inutile, aucun caractère accentués.
- Extended ASCII : ajouts de quelques caractères accentués et de caractères d'affichages.
- ISO 8859-1 (latin1) : pour le français il manque œ, Œ et Ÿ.

Encodage

Un encodage est une manière d'**interpréter** un tableau d'octets.

Exemples :

- ASCII : 8ème bit est inutile, aucun caractère accentués.
- Extended ASCII : ajouts de quelques caractères accentués et de caractères d'affichages.
- ISO 8859-1 (latin1) : pour le français il manque œ, Œ et Ÿ.
- ISO 8859-15 : remplacement de certains caractères par €, Š, š, Ž, ž, Œ, œ et Ÿ.

Encodage

Un encodage est une manière d'**interpréter** un tableau d'octets.

Exemples :

- ASCII : 8ème bit est inutile, aucun caractère accentués.
- Extended ASCII : ajouts de quelques caractères accentués et de caractères d'affichages.
- ISO 8859-1 (latin1) : pour le français il manque œ, Œ et Ÿ.
- ISO 8859-15 : remplacement de certains caractères par €, Š, š, Ž, ž, Œ, œ et Ÿ.
- CP1252 : variante de ISO, utilisé par les vieux Windows.

Encodage

Un encodage est une manière d'**interpréter** un tableau d'octets.

Exemples :

- ASCII : 8ème bit est inutile, aucun caractère accentués.
- Extended ASCII : ajouts de quelques caractères accentués et de caractères d'affichages.
- ISO 8859-1 (latin1) : pour le français il manque œ, Œ et Ÿ.
- ISO 8859-15 : remplacement de certains caractères par €, Š, š, Ž, ž, Œ, œ et Ÿ.
- CP1252 : variante de ISO, utilisé par les vieux Windows.
- UTF-8 : les caractères sont codés sur 1 à 4 octets.

Encodage

Un encodage est une manière d'**interpréter** un tableau d'octets.

Exemples :

- ASCII : 8ème bit est inutile, aucun caractère accentués.
- Extended ASCII : ajouts de quelques caractères accentués et de caractères d'affichages.
- ISO 8859-1 (latin1) : pour le français il manque œ, Œ et Ÿ.
- ISO 8859-15 : remplacement de certains caractères par €, Š, š, Ž, ž, Œ, œ et Ÿ.
- CP1252 : variante de ISO, utilisé par les vieux Windows.
- UTF-8 : les caractères sont codés sur 1 à 4 octets.
- UTF-16, UTF-32 : "mieux" mais incompatibles avec ASCII.

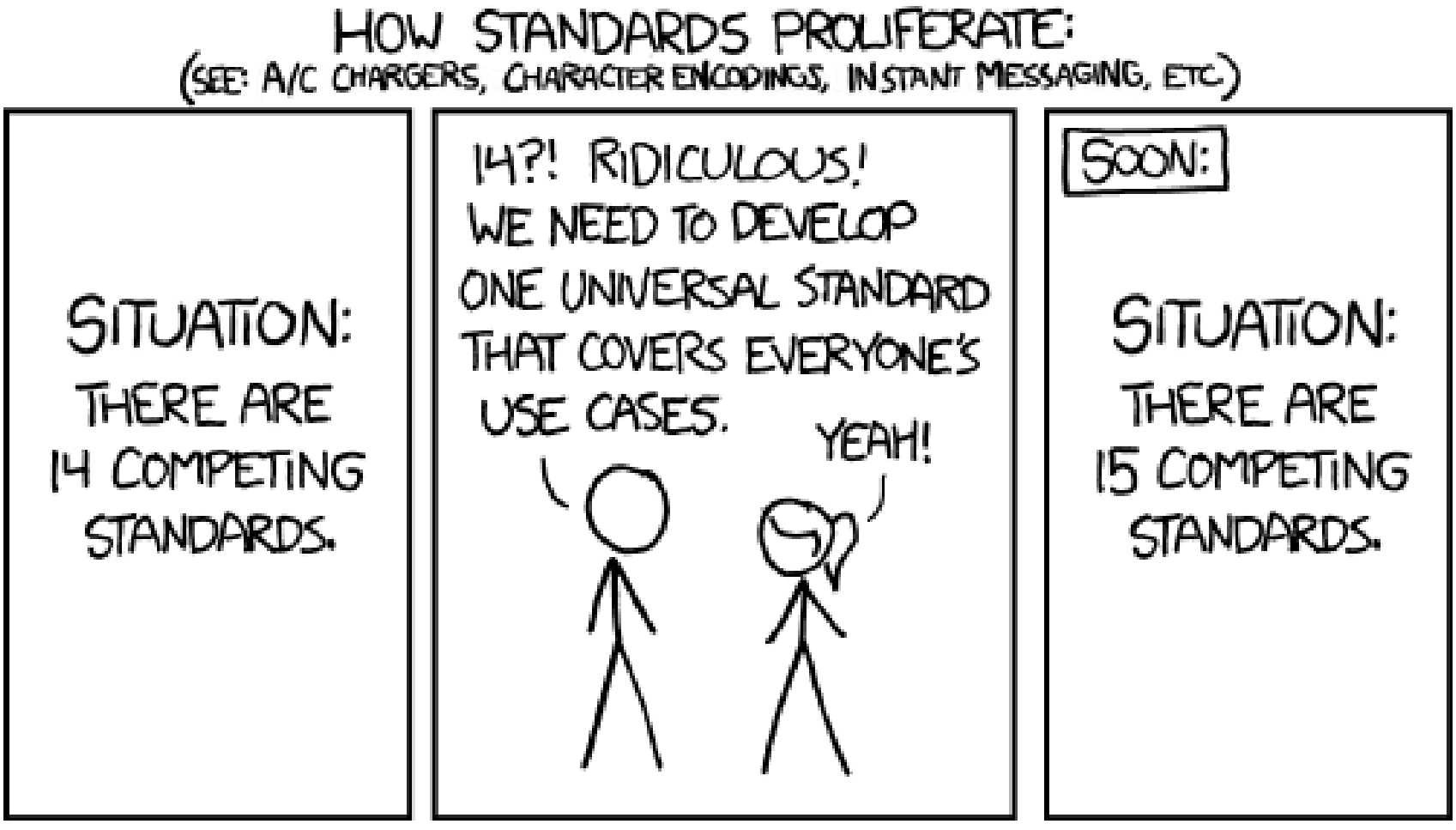
Encodage

Un encodage est une manière d'**interpréter** un tableau d'octets.

Exemples :

- ASCII : 8ème bit est inutile, aucun caractère accentués.
- Extended ASCII : ajouts de quelques caractères accentués et de caractères d'affichages.
- ISO 8859-1 (latin1) : pour le français il manque œ, Œ et Ÿ.
- ISO 8859-15 : remplacement de certains caractères par €, Š, š, Ž, ž, Œ, œ et Ÿ.
- CP1252 : variante de ISO, utilisé par les vieux Windows.
- UTF-8 : les caractères sont codés sur 1 à 4 octets.
- UTF-16, UTF-32 : "mieux" mais incompatibles avec ASCII.
- Beaucoup d'autres...

Mais pourquoi y a-t-il autant d'encodages ?



Pourquoi et comment connaître l'encodage ?

L'encodage est important si l'on accède au système de fichier : un fichier texte est essentiellement une chaîne de caractère.

Pourquoi et comment connaître l'encodage ?

L'encodage est important si l'on accède au système de fichier : un fichier texte est essentiellement une chaîne de caractère.

En bref, on ne peut pas le connaître.

Pourquoi et comment connaître l'encodage ?

L'encodage est important si l'on accède au système de fichier : un fichier texte est essentiellement une chaîne de caractère.

En bref, on ne peut pas le connaître.

- Il n'existe pas de manière standard de déclarer l'encodage.
  - Plusieurs types de fichiers (HTML, XML) peuvent le déclarer en haut du fichier.
    (Cette déclaration n'utilise que les 127 caractères ASCII.)

Pourquoi et comment connaître l'encodage ?

L'encodage est important si l'on accède au système de fichier : un fichier texte est essentiellement une chaîne de caractère.

En bref, on ne peut pas le connaître.

- Il n'existe pas de manière standard de déclarer l'encodage.
  - Plusieurs types de fichiers (HTML, XML) peuvent le déclarer en haut du fichier.
    (Cette déclaration n'utilise que les 127 caractères ASCII.)
  - Les fichiers codés en UTF8 commencent (optionnellement...) par un caractère spécial (qui ne s'affiche pas) qui, dans les fait, assure que le fichier est en UTF8.
    Une mauvaise interprétation affiche ï»¿ .

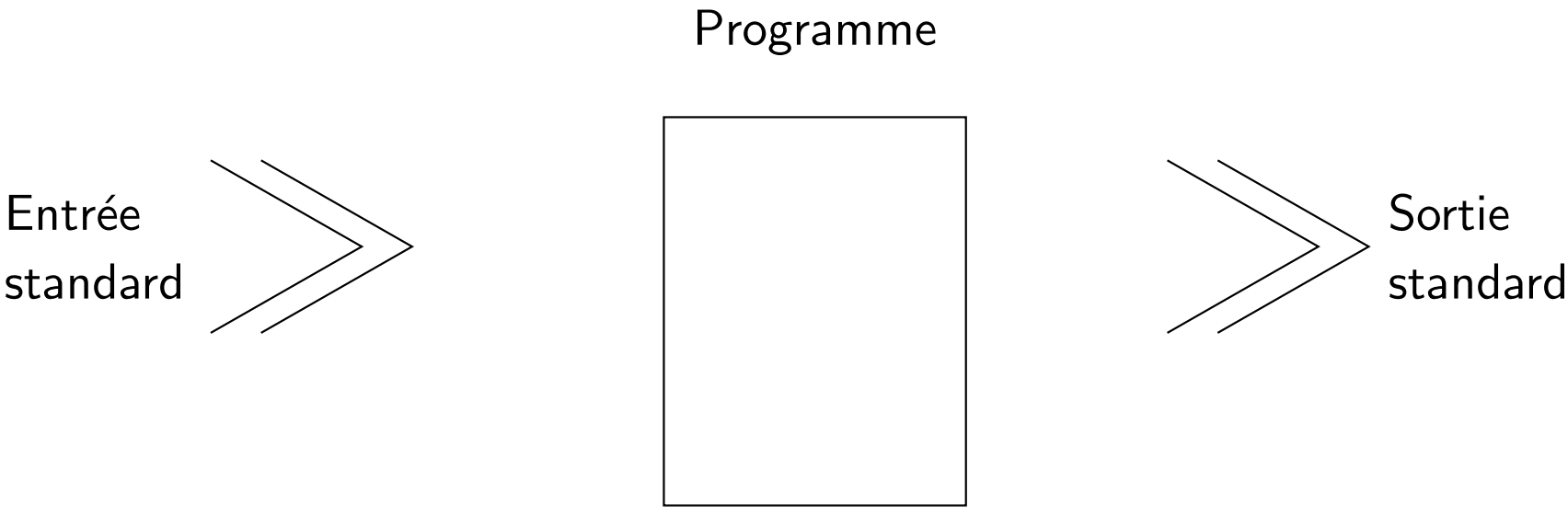
Pourquoi et comment connaître l'encodage ?

L'encodage est important si l'on accède au système de fichier : un fichier texte est essentiellement une chaîne de caractère.

En bref, on ne peut pas le connaître.

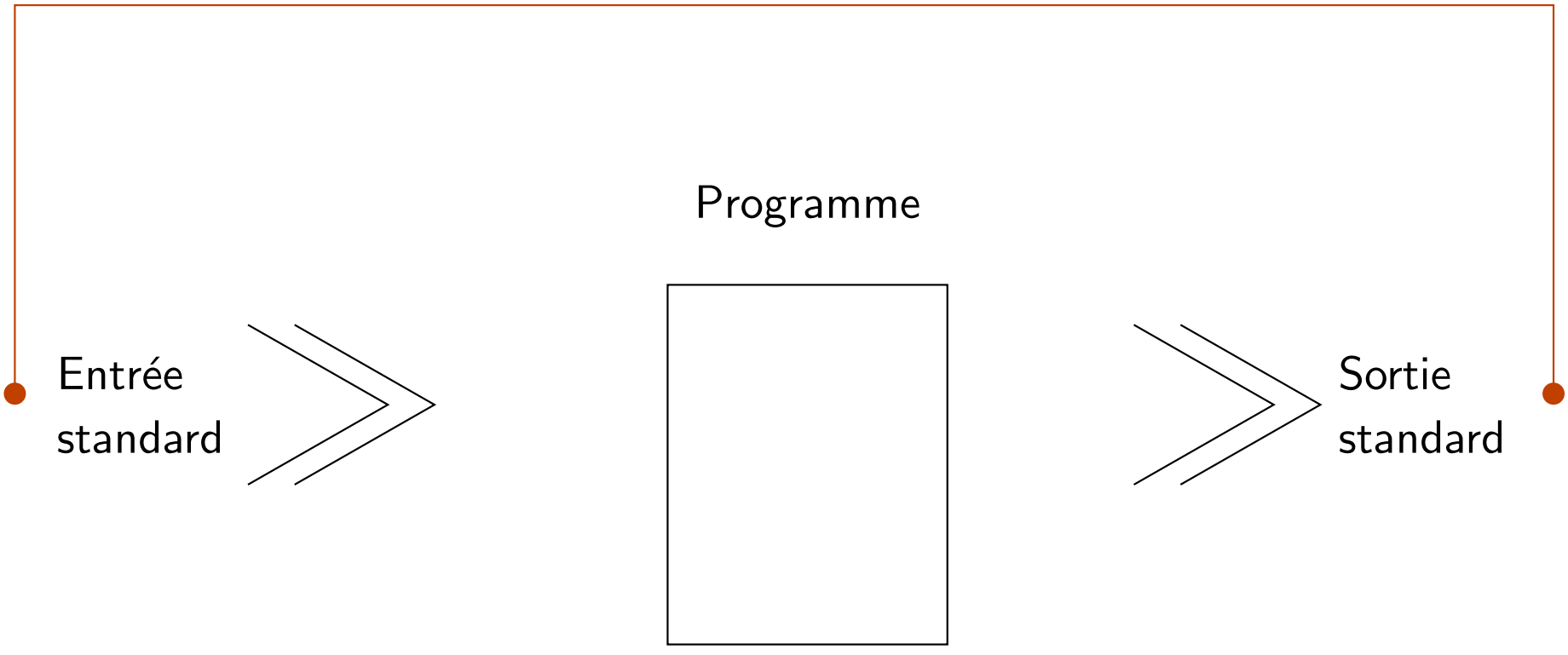
- Il n'existe pas de manière standard de déclarer l'encodage.
  - Plusieurs types de fichiers (HTML, XML) peuvent le déclarer en haut du fichier.
    (Cette déclaration n'utilise que les 127 caractères ASCII.)
  - Les fichiers codés en UTF8 commencent (optionnellement...) par un caractère spécial (qui ne s'affiche pas) qui, dans les fait, assure que le fichier est en UTF8.
    Une mauvaise interprétation affiche ï»¿ .
- Certaines combinaisons de caractères sont interdites dans certains encodages :
  → Le programme `file` sous linux offre une heuristique pour déterminer l'encodage d'un fichier.

Revenons à la programmation

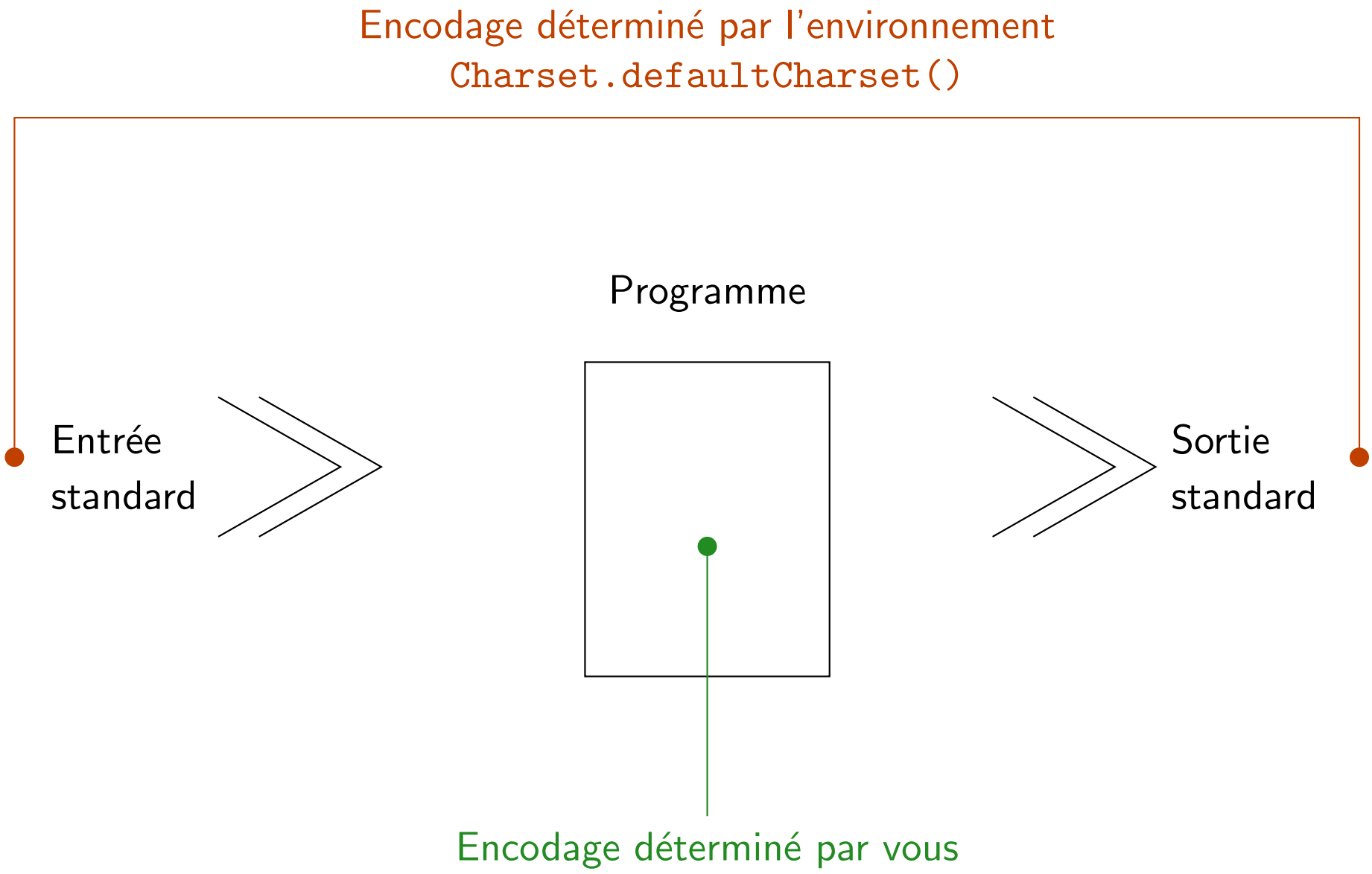


Revenons à la programmation

Encodage déterminé par l'environnement
`Charset.defaultCharset()`



Revenons à la programmation



Entrée standard vers Sortie Standard


```
public static main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println(args[0]);
        System.out.println(s.nextLine());
}
```

```
marsault@ordi : /tmp $  javac Main
marsault@ordi : /tmp $  java Main "a é ï œ ù"
a é ï œ ù
ç̧ç̧ç̧
ç̧ç̧ç̧
```

Entrée standard vers Sortie Standard


```
public static main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println(args[0]);
        System.out.println(s.nextLine());
}
```

```
marsault@ordi : /tmp $  javac Main
marsault@ordi : /tmp $  java Main "a é ï œ ù"
a é ï œ ù
ççç
ççç
```

```
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        javac Main.java
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        java Main "a é ï o ù"
a é ï o ù
ççç
ççç
```

Chaînes de caractères constantes


```
public static void main(String[] args) throws Exception {
    String str1 = new String("> a é ï œ ù");
    System.out.println(str1);
}
```

```
marsault@ordi : /tmp $  javac Main
marsault@ordi : /tmp $  java Main
> a é ï œ ù
```

Chaînes de caractères constantes


```
public static void main(String[] args) throws Exception {
    String str1 = new String("> a é ï œ ù");
    System.out.println(str1);
}
```

```
marsault@ordi : /tmp $  javac Main
marsault@ordi : /tmp $  java Main
> a é ï œ ù
```

```
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        javac Main.java
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        java Main
> a ? ? ? ? ?
```

Chaînes de caractères constantes


```
public static void main(String[] args) throws Exception {
    String str1 = new String("> a é ï œ ù");
    System.out.println(str1);
}
```

```
marsault@ordi : /tmp $  javac Main
marsault@ordi : /tmp $  java Main
> a é ï œ ù
```

```
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        javac Main.java
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        java Main
> a ? ? ? ? ?
```

Pourquoi ça ne fonctionne pas dans le deuxième cas ?

Chaînes de caractères constantes


```
public static void main(String[] args) throws Exception {
    String str1 = new String("> a é ï œ ù");
    System.out.println(str1);
}
```

```
marsault@ordi : /tmp $  javac Main
marsault@ordi : /tmp $  java Main
> a é ï œ ù
```

```
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        javac Main.java
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        java Main
> a ? ? ? ? ?
```

Pourquoi ça ne fonctionne pas dans le deuxième cas ?

$\rightarrow$ L'encodage du fichier source est inchangé. La chaîne constante est donc censée être en UTF8 mais est codée en ISO-8859-1.

Ce qu'on peut faire


```
public static void main(String[] args) throws Exception {
    byte[] tab=
        "> a é ï œ ù".getBytes(Charset.defaultCharset());
    // recalcule la suite d'octets qui représente la chaine
    // dans l'encodage du systeme
```

Ce qu'on peut faire


```
public static void main(String[] args) throws Exception {
    byte[] tab=
        "> a é ï œ ù".getBytes(Charset.defaultCharset());
    // recalcule la suite d'octets qui représente la chaine
    // dans l'encodage du systeme
    String str1 = new String (tab, "UTF8");
    // reinterprete la suite en utilisant l'encodage UTF8
```

Ce qu'on peut faire


```
public static void main(String[] args) throws Exception {
    byte[] tab=
        "> a é ï œ ù".getBytes(Charset.defaultCharset());
    // recalcule la suite d'octets qui représente la chaine
    // dans l'encodage du systeme
    String str1 = new String (tab, "UTF8");
    // reinterprete la suite en utilisant l'encodage UTF8
    System.out.println(str1);
}
```

Ce qu'on peut faire


```
public static void main(String[] args) throws Exception {
    byte[] tab=
        "> a é ï œ ù".getBytes(Charset.defaultCharset());
    // recalcule la suite d'octets qui représente la chaine
    // dans l'encodage du systeme
    String str1 = new String (tab, "UTF8");
    // reinterprete la suite en utilisant l'encodage UTF8
    System.out.println(str1);
}
```

```
marsault@ordi : /tmp $  javac Main.java
marsault@ordi : /tmp $  java Main
> a é ï œ ù
```

```
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        javac Main.java
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        java Main
> a é ï ? ù
```

Ce qu'il vaut mieux faire


```
public static void main(String[] args) throws Exception {
    String str1 = new String("> a é ï œ ù");
    System.out.println(str1);
}
```

Ce qu'il vaut mieux faire


```
public static void main(String[] args) throws Exception {
    String str1 = new String("> a é ï œ ù");
    System.out.println(str1);
}
```

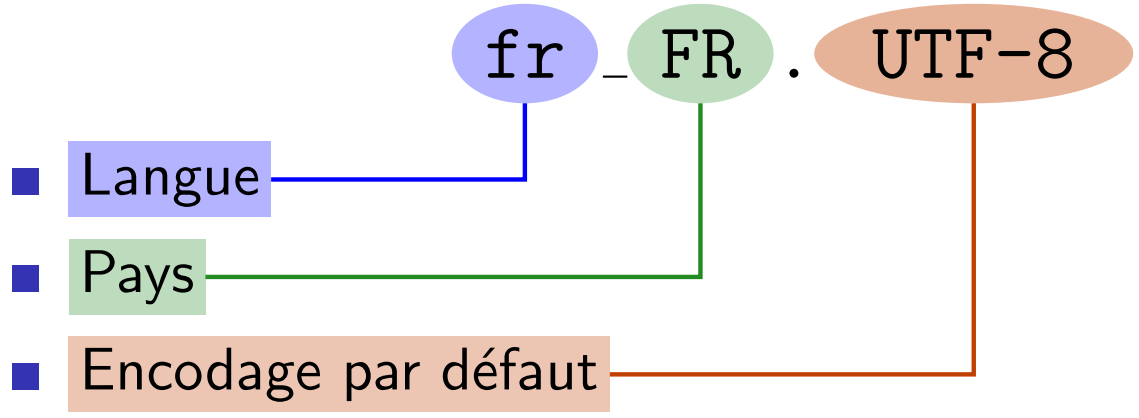
```
marsault@ordi : /tmp $  javac -encoding utf8 Main.java
marsault@ordi : /tmp $  java Main
> a é ï œ ù
```

```
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        javac -encoding utf8 Main.java
marsault@ordi : /tmp $  LC_ALL=fr_FR.ISO-8859-1
                        java Main
> a é ï ? ù
```

Locale

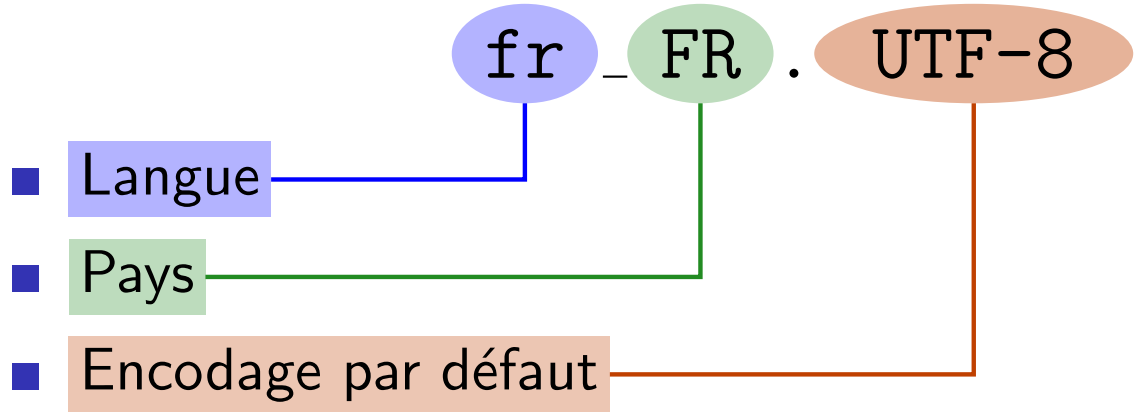
- Langue
- Pays
- Encodage par défaut

Locale



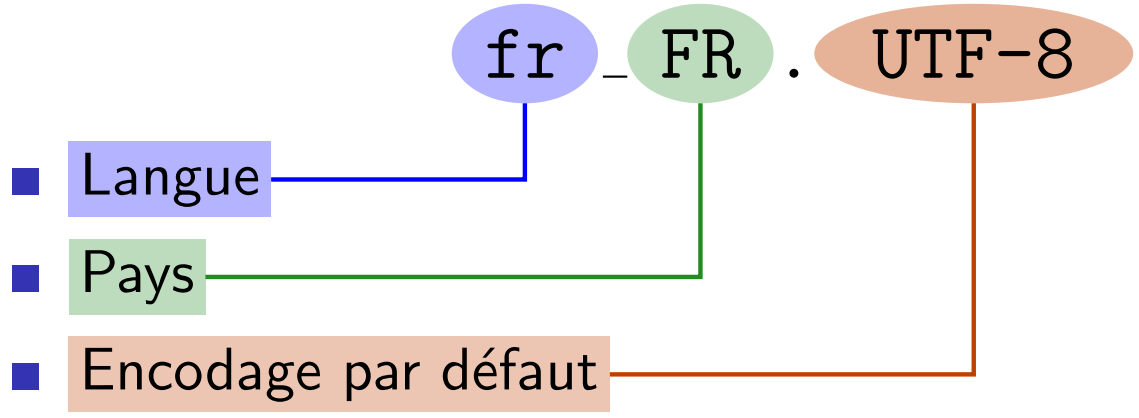

```
static public void main (String[] args) {
    Locale defaultLocale = Locale.getDefault();
```

Locale




```
static public void main (String[] args) {
    Locale defaultLocale = Locale.getDefault();
    String codeLangage = defaultLocale.getLanguage();
    // "fr" pour français
    String codePays = defaultLocale.getCountry();
    // "FR" pour France
```

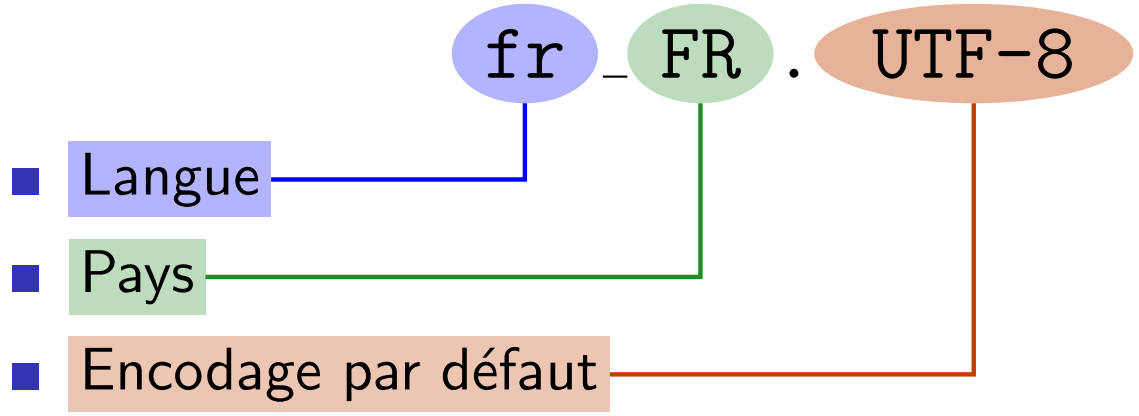
Locale




```
static public void main (String[] args) {
    Locale defaultLocale = Locale.getDefault();
    String codeLangage = defaultLocale.getLanguage();
    // "fr" pour français
    String codePays = defaultLocale.getCountry();
    // "FR" pour France

    Locale anglais = Locale.ENGLISH;
```

Locale




```
static public void main (String[] args) {
    Locale defaultLocale = Locale.getDefault();
    String codeLangage = defaultLocale.getLanguage();
    // "fr" pour français
    String codePays = defaultLocale.getCountry();
    // "FR" pour France

    Locale anglais = Locale.ENGLISH;
    Locale quebecquois = new Locale ("fr", "CA");
}
```

Outline

Fichier et répertoire : File


```
//Constructor
public File(String pathname);
public File(String parent, String child);
public File(File parent, String child);
```

Fichier et répertoire : File


```
//Constructor
public File(String pathname);
public File(String parent, String child);
public File(File parent, String child);

//Creation (renvoie false si existe,
//          lance une exception si chemin invalide)
public boolean createNewFile(); //fichier vide
public boolean mkdir(); //repertoire
```

Fichier et répertoire : File


```
//Constructor
public File(String pathname);
public File(String parent, String child);
public File(File parent, String child);

//Creation (renvoie false si existe,
//          lance une exception si chemin invalide)
public boolean createNewFile(); //fichier vide
public boolean mkdir(); //repertoire
public boolean mkdirs(); //repertoires
```

Fichier et répertoire : File

```
//Constructor
public File(String pathname);
public File(String parent, String child);
public File(File parent, String child);

//Creation (renvoie false si existe,
//          lance une exception si chemin invalide)
public boolean createNewFile(); //fichier vide
public boolean mkdir(); //repertoire
public boolean mkdirs(); //repertoires
public boolean delete();
```

Fichier et répertoire : File

```
//Constructor
public File(String pathname);
public File(String parent, String child);
public File(File parent, String child);

//Creation (renvoie false si existe,
//          lance une exception si chemin invalide)
public boolean createNewFile(); //fichier vide
public boolean mkdir(); //repertoire
public boolean mkdirs(); //repertoires
public boolean delete();

// Tests
public boolean isDirectory();
public boolean isFile();
public boolean exists();
```

Fichier et répertoire : File (2)


```
//Position
public String getPath(); //nom complet comme fourni
public String getCanonicalPath(); //nom depuis la racine
```

Fichier et répertoire : File (2)


```
//Position
public String getPath(); //nom complet comme fourni
public String getCanonicalPath(); //nom depuis la racine
public String getName(); //nom "top-level"
public String getParent() //repertoire contenant
```

Fichier et répertoire : File (2)


```
//Position
public String getPath(); //nom complet comme fourni
public String getCanonicalPath(); //nom depuis la racine
public String getName(); //nom "top-level"
public String getParent() //repertoire contenant

// Droits
public boolean canRead() //canWrite(), canExecute()
public boolean setReadable(boolean b) //setWritable,..
public boolean setReadable(boolean b, boolean ownerOnly);
```

Fichier et répertoire : File (2)


```
//Position
public String getPath(); //nom complet comme fourni
public String getCanonicalPath(); //nom depuis la racine
public String getName(); //nom "top-level"
public String getParent() //repertoire contenant

// Droits
public boolean canRead() //canWrite(), canExecute()
public boolean setReadable(boolean b) //setWritable,..
public boolean setReadable(boolean b, boolean ownerOnly);

// Parcourir (renvoie null si n'est pas un repertoire)
public String[] list() // noms des éléments contenus
public File[]  listFiles() // 'File' contenues
```

Fichier et répertoire : File (2)


```
//Position
public String getPath(); //nom complet comme fourni
public String getCanonicalPath(); //nom depuis la racine
public String getName(); //nom "top-level"
public String getParent() //repertoire contenant

// Droits
public boolean canRead() //canWrite(), canExecute()
public boolean setReadable(boolean b) //setWritable,..
public boolean setReadable(boolean b, boolean ownerOnly);

// Parcourir (renvoie null si n'est pas un repertoire)
public String[] list() // noms des éléments contenus
public File[]  listFiles() // 'File' contenues

pub static final String pathSeparator = "/"; //Linux/Unix
                                      = "\"; //Windows
```

Création de quelques fichiers


```
public static void main(String[] args) throws Exception {
    File f = new File("tmp");
    f.mkdir();
```

Création de quelques fichiers


```
public static void main(String[] args) throws Exception {
    File f = new File("tmp");
    f.mkdir();
    //File f2 = new File(f,"tmp2/test")
    //n'est pas interoperable
    File f2 = new File(f, "tmp"+
                          File.pathSeparator+
                          "test");
      // ou = new File (f, new File("tmp2", "test));
```

Création de quelques fichiers


```
public static void main(String[] args) throws Exception {
    File f = new File("tmp");
    f.mkdir();
    //File f2 = new File(f,"tmp2/test")
    //n'est pas interoperable
    File f2 = new File(f, "tmp"+
                          File.pathSeparator+
                          "test");
      // ou = new File (f, new File("tmp2", "test));

    //f2 represente tmp/tmp2/test
```

Création de quelques fichiers


```
public static void main(String[] args) throws Exception {
    File f = new File("tmp");
    f.mkdir();
    //File f2 = new File(f,"tmp2/test")
    //n'est pas interoperable
    File f2 = new File(f, "tmp"+
                          File.pathSeparator+
                          "test");
      // ou = new File (f, new File("tmp2", "test));

    //f2 represente tmp/tmp2/test
    //f2.createNewFile() lancerait une exception:
    // tmp/tmp2 n'existe pas

    f2.getParentFile().mkdir;
    f2.createNewFile()
    String tab = f2.getParentFile().list(); // "test"
}
```

Lecture de fichier : BufferedReader/FileReader


```
public static void main(String[] args) throws Exception {

    File file = ... ;
    BufferedReader in =
            new BufferedReader(new FileReader(file));
    while (in.ready())
        System.out.println(in.readLine());
}
```

Écriture de fichier : BufferedWriter/FileWriter


```
public static void main(String[] args) throws Exception {

    File file = ... ;
    BufferedWriter out
            = new BufferedWriter(new FileWriter(file));

    out.write("quelques caracteres");
```

Écriture de fichier : BufferedWriter/FileWriter


```
public static void main(String[] args) throws Exception {

    File file = ... ;
    BufferedWriter out
            = new BufferedWriter(new FileWriter(file));

    out.write("quelques caracteres");
    out.newLine(); //bien mieux que write('\n').
        //La chaine "fin de ligne" dépend du systeme
        //'\n' sous Linux/Unix, '\r\n' sous windows.
        // Voir System.getProperty("line.separator")
        // et '%n' dans String.format().
```

Écriture de fichier : BufferedWriter/FileWriter


```
public static void main(String[] args) throws Exception {

    File file = ... ;
    BufferedWriter out
            = new BufferedWriter(new FileWriter(file));

    out.write("quelques caracteres");
    out.newLine(); //bien mieux que write('\n').
        //La chaine "fin de ligne" dépend du systeme
        //'\n' sous Linux/Unix, '\r\n' sous windows.
        // Voir System.getProperty("line.separator")
        // et '%n' dans String.format().

    out.write("lës ênnùïs @rrîvent");
```

Écriture de fichier : BufferedWriter/FileWriter


```
public static void main(String[] args) throws Exception {

    File file = ... ;
    BufferedWriter out
            = new BufferedWriter(new FileWriter(file));

    out.write("quelques caracteres");
    out.newLine(); //bien mieux que write('\n').
        //La chaine "fin de ligne" dépend du systeme
        //'\n' sous Linux/Unix, '\r\n' sous windows.
        // Voir System.getProperty("line.separator")
        // et '%n' dans String.format().

    out.write("lës ênnùïs @rrîvent");
    //compiler avec java -encoding utf8 etc.
}
```

Écrire avec un encodage


```
public static void ecritEnIso(File file, String data) {
    FileOutputStream out = new FileOutputStream(file);
    OutputStreamWriter writer =
            new OutputStreamWriter(out,
                    Charset.forName("ISO-8859-1"));
```

Écrire avec un encodage


```
public static void ecritEnIso(File file, String data) {
    FileOutputStream out = new FileOutputStream(file);
    OutputStreamWriter writer =
            new OutputStreamWriter(out,
                    Charset.forName("ISO-8859-1"));
    writer.write(data);
    //writer.newLine() n'existe pas
    writer.write(System.getProperty("line.separator"));
}
```

Outline

Quésako ?

Sérialisation = "mettre en série"

- Transformer une instance quelconque en `String`.
- Transformer une `String` en instance (du bon type).

Quésako ?

Sérialisation = "mettre en série"

- Transformer une instance quelconque en `String`.
- Transformer une `String` en instance (du bon type).

Permet

- d'enregistrer des instances directement dans un fichier puis de les rouvrir :
  - précalcul (dictionnaires, etc.) ;
  - sauvegarde pour débug ;

Quésako ?

Sérialisation = "mettre en série"

- Transformer une instance quelconque en `String`.
- Transformer une `String` en instance (du bon type).

Permet

- d'enregistrer des instances directement dans un fichier puis de les rouvrir :
  - précalcul (dictionnaires, etc.) ;
  - sauvegarde pour débug ;
- d'échanger directement des instances par le réseau.

Exemple de `Serializable`

Implémenter `Serializable`

Chaque attribut doit

- soit être déclaré `transient` (à ne pas enregistrer) ;
- soit être l'instance d'une classe qui implémente `Serializable` ;
- soit être un type primitif.

Exemple de `Serializable`

Implémenter `Serializable`

Chaque attribut doit

- soit être déclaré `transient` (à ne pas enregistrer) ;
- soit être l'instance d'une classe qui implémente `Serializable` ;
- soit être un type primitif.

```
public class maClasse implements Serializable {
    private String nom;//OK:implémente Serializable
```

Exemple de `Serializable`

Implémenter `Serializable`

Chaque attribut doit

- soit être déclaré `transient` (à ne pas enregistrer) ;
- soit être l'instance d'une classe qui implémente `Serializable` ;
- soit être un type primitif.

```
public class maClasse implements Serializable {
    private String nom;//OK:implémente Serializable
    private int id;//OK:type primitif
```

Exemple de Serializable

Implémenter Serializable

Chaque attribut doit

- soit être déclaré `transient` (à ne pas enregistrer) ;
- soit être l'instance d'une classe qui implémente `Serializable` ;
- soit être un type primitif.

```
public class maClasse implements Serializable {
    private String nom;//OK:implémente Serializable
    private int id;//OK:type primitif
    public Pile<String> classe pile;//Oups, Pile n'est
                                    //pas serialisable
```

Rendons `Pile` sérialisable


```
public class Pile<E>
{

    private static class Cellule<F>
    {
        private F val;
        private Cellule<F> suivant;
        ...
    }
    Cellule<E> hautDePile;
    ...
}
```

Rendons `Pile` sérialisable


```
public class Pile<E>
      implements Serializable {

    private static class Cellule<F>
    {
        private F val;
        private Cellule<F> suivant;
        ...
    }
    Cellule<E> hautDePile;
    ...
}
```

Rendons `Pile` sérialisable


```
public class Pile<E>
      implements Serializable {

    private static class Cellule<F>
          implements Serializable {
        private F val;
        private Cellule<F> suivant;
        ...
    }
    Cellule<E> hautDePile;
    ...
}
```

Rendons `Pile` sérialisable


```
public class Pile<E>
      implements Serializable {

    private static class Cellule<F extends Serializable>
          implements Serializable {
        private F val;
        private Cellule<F> suivant;
        ...
    }
    Cellule<E> hautDePile;
    ...
}
```

Rendons `Pile` sérialisable


```
public class Pile<E extends Serializable>
      implements Serializable {

    private static class Cellule<F extends Serializable>
          implements Serializable {
        private F val;
        private Cellule<F> suivant;
        ...
    }
    Cellule<E> hautDePile;
    ...
}
```

Exemple de Serializable (2)


```
public class maClasse implements Serializable {
    private String nom;//OK
    private int id;//OK
    public Pile<String> classe pile;//OK
    ...
}
```

Exemple de Serializable (2)


```
public class maClasse implements Serializable {
    private String nom;//OK
    private int id;//OK
    public Pile<String> classe pile;//OK
    ...
}

public static void main(String [] args) {
    maClasse vm = new maClasse("Victor",1);
    maClasse ad = new maClasse("Aldric",0);
    ad.pile.add("Cours 1");    ad.pile.add("Cours 2");
```

Exemple de Serializable (2)


```
public class maClasse implements Serializable {
    private String nom;//OK
    private int id;//OK
    public Pile<String> classe pile;//OK
    ...
}

public static void main(String [] args) {
    maClasse vm = new maClasse("Victor",1);
    maClasse ad = new maClasse("Aldric",0);
    ad.pile.add("Cours 1");    ad.pile.add("Cours 2");
    FileOutputStream fout =
          new FileOutputStream("tmp");
    ObjectOutputStream out = new ObjectOutputStream(fout);
    out.writeObject(vm);   out.writeObject(ad);
    out.close();
    fileOut.close();
}
```

Exemple de Serializable (3)


```
public static void main(String [] args) {
    SerialTest vm2,ad2;
```

Exemple de Serializable (3)


```
public static void main(String [] args) {
    SerialTest vm2,ad2;
    FileInputStream fin = new FileInputStream("tmp");
```

Exemple de Serializable (3)


```
public static void main(String [] args) {
    SerialTest vm2,ad2;
    FileInputStream fin = new FileInputStream("tmp");
    ObjectInputStream in = new ObjectInputStream(fin);
```

Exemple de Serializable (3)


```
public static void main(String [] args) {
    SerialTest vm2,ad2;
    FileInputStream fin = new FileInputStream("tmp");
    ObjectInputStream in = new ObjectInputStream(fin);
    vm2 = (SerialTest) in.readObject();
    ad2 = (SerialTest) in.readObject();
```

Exemple de Serializable (3)


```
public static void main(String [] args) {
    SerialTest vm2,ad2;
    FileInputStream fin = new FileInputStream("tmp");
    ObjectInputStream in = new ObjectInputStream(fin);
    vm2 = (SerialTest) in.readObject();
    ad2 = (SerialTest) in.readObject();
    in.close(); fin.close();

    System.out.println(ad2); //affiche l'addresse de ad2
    for (String s: ad2.pile)
      System.out.println(s); //affiche "Cours 1" et "Cours 2"
```

Caveat

Implémenter `Serializable`

Chaque attribut doit

- soit être déclaré `transient` (à ne pas enregistrer) ;
- soit être l'instance d'une classe qui implémente `Serializable` ;
- soit être un type primitif.

Caveat

Implémenter `Serializable`

Chaque attribut doit

- soit être déclaré `transient` (à ne pas enregistrer) ;
- soit être l'instance d'une classe qui implémente `Serializable` **À L'ÉXECUTION** ;
- soit être un type primitif.

Déclarer qu'une classe implémente `Serializable` ne provoquera jamais aucune erreur à la compilation.

Caveat

Implémenter `Serializable`

Chaque attribut doit

- soit être déclaré `transient` (à ne pas enregistrer) ;
- soit être l'instance d'une classe qui implémente `Serializable` **À L'ÉXECUTION** ;
- soit être un type primitif.

Déclarer qu'une classe implémente `Serializable` ne provoquera jamais aucune erreur à la compilation.

En particulier, il n'est jamais nécessaire de déclarer les variables de type générique comme étant sérialisable :
`Pile<E extends Serializable>` `extends Serializable` .