# 420-111-SF - ALGORITHMIQUE ET PROGRAMMATION

# TRAVAIL PRATIQUE #2

## SPÉCIFICATIONS DE REMISE

- Pondération : **20 %**
- Réalisation **: en équipe de 2 étudiants-es**

**REMISE 1 –** vendredi 30 novembre 23 h

- Remise individuelle de la grille d'évaluation de l'équipe complétée par chaque étudiant.

**REMISE 2 –** vendredi 7 décembre 23 h

- Remise individuelle de la grille d'évaluation de l'équipe complétée par chaque étudiant;
- Votre projet Blue J nommé **matricule1_matricule2_TP2.zip;**
- Le document TP2-ListeControle.docx dûment rempli.

## ÉVALUATION

| | |
|---|---|
| **Partie A*** | **20** |
| Fonction A + tests fonctionnels | 7 |
| Fonction B + tests fonctionnels | 8 |
| Fonction C + tests fonctionnels | 2 |
| Fonction D + tests fonctionnels | 3 |
| **Partie B*** | **15** |
| Fonction A + tests fonctionnels | 7 |
| Fonction B + tests fonctionnels | 8 |
| **Partie C*** | **15** |
| Fonction A + tests fonctionnels | 10 |
| Fonction B + tests fonctionnels | 5 |
| **Partie D*** | **10** |
| Fonction C + tests fonctionnels | 5 |
| Fonction D + tests fonctionnels | 5 |
| **Qualité du code**** | **30** |
| **Professionnalisme du travail (remises, liste contrôle, complétion, qualité générale…)** | **10** |

* Si aucun test n'est réalisé, ou en nombre insuffisant et inacceptable, la partie du travail évaluée (A, B, C ou D) ne peut se voir attribuer une note supérieure à 50% (voir barème page suivante).

** Un travail qui n'implémente pas adéquatement les parties A, B, C et D ne peut se voir attribuer une note supérieure à 60% pour cet élément d'évaluation.

Conformément à la politique concernant la qualité du français écrit, le travail est sujet à une pénalité pouvant aller jusqu'à 20% du total des points attribuables.

**Dans le cadre d'un travail d'équipe, un ajustement individuel à la note d'équipe peut être apporté.**

**Si le professeur le juge à propos, tout étudiant(e) pourra être convoqué(e) à une rencontre d'évaluation pour vérifier son degré d'acquisition des connaissances et d'appréhension de la solution proposée.**

Chaque élément d'évaluation est coté selon la grille et le barème suivants :

| Cotes | Barème | Description |
|---|---|---|
| A+<br>A<br>A- | 100<br>95<br>90 | Très professionnel : la plupart des éléments du travail démontrent une maîtrise supérieure de la compétence et quelques éléments additionnels ont été réalisés de manière satisfaisante. |
| B+<br>B<br>B- | 85<br>80<br>75 | Professionnel : tous les éléments du travail ont été réalisés de façon au moins satisfaisante et quelques éléments démontrent une maîtrise supérieure des éléments de la compétence. |
| C+<br>C<br>C- | 70<br>65<br>60 | Moyen : la plupart des éléments requis du travail ont été réalisés de façon satisfaisante, toutefois certains éléments n'ont pas été réalisés ou encore ne le sont pas de façon satisfaisante. |
| D | 50 | Médiocre : certains éléments du travail n'ont pas été réalisés de façon satisfaisante. |
| E | 30 | Inacceptable : plusieurs éléments n'ont pas été réalisés ou encore ont été réalisés de façon nettement insatisfaisante. |
| Z | 0 | Irrecevable : le travail n'a pas été réalisé ou encore ne répond pas aux critères minimaux d'acceptation. |

Se référer à la section « **Évaluation** » du plan de cours pour plus de détails sur l'évaluation des travaux.

## PRÉSENTATION

Nous allons construire une application d'encryption.

Dans cette application, l'utilisateur aura le choix entre encrypter un message texte et décrypter un message texte encrypté. Lors de l'encryption, le message à encrypter sera transformé en une séquence binaire (tableau) et cette dernière sera ensuite encryptée. Le résultat final de cette encryption sera encodé en hexadécimal (plus compact).

Le chemin inverse sera effectué pour déchiffrer le message.

*Merci à notre collègue François Gagnon pour la conception de ce travail pratique!*

## CONSIGNES GÉNÉRALES

**IMPORTANT!**

- Des questionnements sur les différentes conversions à programmer?
    - Vous pouvez puiser dans votre matériel du cours système d'exploitation…
    - Vous pouvez consulter quantité de ressources. En voici deux :
        - Conversion du décimal en binaire : 2 méthodes
        - Bases décimale, binaire et hexadécimale
    - Référez-vous au complément d'information en annexe de ce document.
- Vous n'avez pas le droit d'utiliser la classe String.
- Vous n'avez pas le droit d'utiliser des fonctions de java déjà existantes.
- Vous devez seulement vous assurer de la fiabilité des fonctions.
  Vous n'avez pas à gérer et tester la robustesse des fonctions.
- Vous devez respecter les standards et bonnes pratiques de programmation vus en classe.
- Il faut, au besoin, découper votre code et définir de nouvelles fonctions que vos fonctions peuvent appeler.

## PARTIE A

Pour l'instant, nous allons programmer les fonctionnalités de bases pour nous préparer aux étapes de chiffrement.

**Mandat**

- Vous devez implémenter 4 fonctions dans la classe TP2PartieA.
- Vous devez aussi ajouter 2 tests unitaires pour chacune des fonctions.

**Consignes**

- Lisez les tests pour mieux comprendre ce que chaque fonction doit faire.
- Ajouter vos fonctions de tests complètement au début de la classe de tests.

**Fichiers**

- TP2PartieA.java
  On vous fournit deux fonctions (transformASCII_IntToChar et transformASCII_CharToInt) que vous pouvez utiliser si désiré.
- TP2PartieATest.java
  Pour chaque fonction, on vous fournit plusieurs tests unitaires dans la classe TP2PartieATest.

**Fonctions à implémenter :**

A. `static int convertFromBinaryToInt(char[] binarySequence)`

Cette fonction reçoit une séquence binaire (binarySequence) qui représente un entier encodé en binaire non-signé. L'objectif de cette fonction est de trouver la valeur décimale (en base 10) de la séquence binaire donné.

B. `static char[] convertFromIntToBinary(int value, int resultSize)`

Cette fonction est l'inverse de A. Elle reçoit un nombre décimal (value) et une dimension. Elle doit produire la séquence binaire représentant value. L'argument resultSize indique la taille que la réponse doit avoir.

C. `static char[] convertCharToBinary(char c)`

Cette fonction reçoit un caractère (c) en paramètre. Elle doit transformer ce caractère en entier (selon la table ASCII) puis encoder l'entier en binaire sur 8 bits.

D. `static char convertBinaryToChar(char[] binarySequence)`

Cette fonction est l'inverse de C. Elle reçoit une séquence binaire (8 bits) représentant un caractère ASCII et elle doit retourner le caractère correspondant. On transfère d'abord la séquence binaire en nombre entier, puis on trouve le caractère correspondant selon la table ASCII.

## PARTIE B

**Mandat**

- Vous devez implémenter 2 fonctions dans la classe TP2PartieB.
- Vous devez aussi réaliser tous les tests unitaires pour les 2 fonctions, au moins 4 tests unitaires pour chacune des fonctions. N'hésitez pas à en ajouter pour vous assurer un niveau de confiance en la fiabilité de vos fonctions….

**Consignes**

- Vous devez utiliser des fonctions réalisées dans la partie A.

A. `static char[] convertCharArrayToBinary(char[] charArray)`

Cette fonction a comme objectif d'encoder une chaîne de caractères (`charArray`) en binaire (selon la table ASCII). Chaque caractère s'encode sur 8 bits. Dans l'exemple ci-dessous, le paramètre tableau est de longueur 2 mais, bien sûr, il peut être de n'importe quelle longueur.

Par exemple:

Paramètre: char[] c = {'1','C'};
Valeur de retour: char[] expectedAnswer = {'0','0','1','1','0','0','0','1','0','1','0','0','0','0','1','1'};

B. `static char[] convertBinaryToCharArray(char[] binaryArray)`

Cette fonction est l'inverse de A. Elle reçoit une séquence binaire (des octets) qu'elle transforme en séquence de caractères selon l'encodage de la table ASCII.

Par exemple:

Paramètre: char[] binaryArray={'0', '1','0', '0','0', '0','0', '1','0', '0','1', '0','1', '1','0', '1'}
Valeur de retour: char[] expected={'A','-'}

## PARTIE C

**Mandat**

- Vous devez implémenter 2 fonctions dans la classe TP2PartieC.
- Vous devez aussi réaliser tous les tests unitaires pour les 2 fonctions, au moins 4 tests unitaires pour chacune des fonctions. N'hésitez pas à en ajouter pour vous assurer un niveau de confiance en la fiabilité de vos fonctions….

**Consignes**

- Vous avez à utiliser des fonctions réalisées dans les parties précédentes.
- Voir la section Chiffrement XOR page suivante.

**Fichiers**

- TP2PartieC.java
  On vous fournit une fonction mainCipher. Vous ne pouvez pas modifier cette fonction, exceptée pour enlever les commentaires, lorsque les fonctions seront programmées et testées, et que vous voudrez exécuter le programme graphique.
- EncryptorPartieC.java et GUI.java
  On vous fournit un programme de chiffrement avec une petite interface graphique (sans validations) qui utilise les fonctions de cette partie. Vous pouvez l'exécuter en appelant la fonction execute() de la classe EncryptorPartieC, qui se charge d'appeler la fonction mainCipher() de la classe TP2PartieC.

A. **char**[] encryptXOR(**char**[] charMsg, **char**[] charKey)

   Cette fonction reçoit un tableau de caractères représentant le message à chiffrer (msg) et un second tableau de caractères représentant la clé de chiffrement (key). Elle retourne un tableau binaire contenant le message chiffré selon l'algorithme de chiffrement XOR.

B. **char**[] decryptXOR(**char**[] binaryMsg, **char**[] charKey)

   Cette fonction reçoit un tableau binaire représentant le message chiffré (binaryMsg) et un tableau de caractères représentant la clé de chiffrement (key). Elle retourne un tableau de caractères contenant le message clair déchiffré selon la clé donnée et l'algorithme XOR.

**Chiffrement XOR :**

Pour chiffrer un message avec XOR, on transforme d'abord les séquences de caractères (message et clé) en encodage binaire. On applique l'opération XOR à chaque paire de bits (message et clé), pour obtenir une nouvelle séquence binaire. Au besoin, on utilise plusieurs copies de la clé. L'opération XOR (ou-exclusif) est un opérateur booléen qui retourne vrai si et seulement si exactement une des deux opérandes est vraie. On suit la convention disant que '1' = vrai et '0' = faux.

Exemple :

- message : « test »
- clé : « ab »

| **Message** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| t | | | | | | | | e | | | | | | | | s | | | | | | | | t | | | | | | | |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

| **Clé*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | | | | | | | | b | | | | | | | | a | | | | | | | | b | | | | | | | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

*On répète la clé autant de fois que nécessaire.

| **Résultat (binaire)** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

| **Résultat (hexa)** | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 0 | 7 | 1 | 2 | 1 | 6 |

Pour déchiffrer un message avec XOR, on transforme la clé (séquence de caractères) en séquence binaire (**le message codé est déjà en binaire**). On applique l'opération XOR à chaque paire de bit (comme pour le chiffrement) pour obtenir une séquence binaire. En transformant cette séquence binaire en séquence de caractères, on retrouve le message initial en clair.

## PARTIE D

**Mandat**

- Vous devez implémenter 2 fonctions dans la classe TP2PartieD.
- Vous devez aussi réaliser tous les tests unitaires pour les 2 fonctions, au moins 4 tests unitaires pour chacune des fonctions. N'hésitez pas à en ajouter pour vous assurer un niveau de confiance en la fiabilité de vos fonctions….

**Consignes**

- Vous avez à utiliser des fonctions réalisées dans les parties précédentes.

**Fichiers**

- TP2PartieD.java
  On vous fournit une fonction mainCipher. Vous ne pouvez pas modifier cette fonction, exceptée pour enlever les commentaires, lorsque les fonctions seront programmées et testées, et que vous voudrez exécuter le programme graphique.
- EncryptorPartieD.java

  On vous fournit un programme de chiffrement avec une petite interface graphique (sans validations) qui utilise les fonctions de cette partie. Vous pouvez l'exécuter en appelant la fonction execute() de la classe EncryptorPartieD, qui se charge d'appeler la fonction mainCipher() de la classe TP2PartieD.

A. `char[] convertBinaryArrayToHexArray(char[] binaryArray)`

Cette fonction reçoit un tableau binaire représentant des caractères. Elle retourne la séquence binaire encodée en hexadécimal.

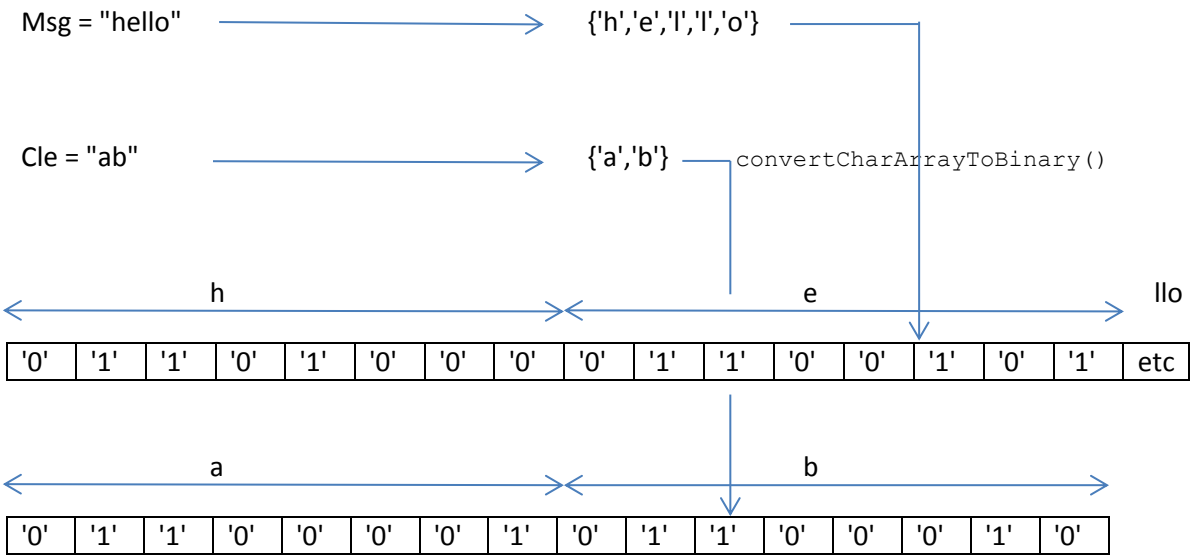
B. `char[] convertHexArrayToBinaryArray(char[] hexArray)`

Cette fonction est l'inverse de A. Elle reçoit une séquence hexadécimal dans un tableau qu'elle doit convertir en séquence binaire.

## ANNEXE

# Encryption – Les étapes

Pour comprendre cette démonstration, il faut se représenter un caractère sur un octet (8 bits).




XOR cellule par cellule :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| '0' | '0' | '0' | '0' | '1' | '0' | '0' | '1' | '0' | '0' | '0' | '0' | '0' | '1' | '1' | '1' | etc |

# Conversion de binaire en hexadécimal

1. Convertir le binaire en décimal:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| '0' | '0' | '0' | '0' | '1' | '0' | '0' | '1' | '0' | '0' | '0' | '0' | '0' | '1' | '1' | '1' | |
| 9 | | | | | | | | 7 | | | | | | | | etc |

2. Puis convertir le décimal en hexadécimal:

| | | |
|---|---|---|
| 9 | 7 | etc |
| '0' \| '9' | '0' \| '7' | |

## Correspondance décimal hexadécimal

| Décimal | Hexadécimal |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | A |
| 11 | B |
| 12 | C |
| 13 | D |
| 14 | E |
| 15 | F |

## Table de vérité du OU Exclusif

| A | B | $A \oplus B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |