# Compte-rendu de TP : Implantation du code CRC en Java

MACIA Mathys & JRIRI Laila

26 janvier 2026

## Table des matières

# 1 Rappel du sujet

L'objectif de ce travail pratique est de concevoir et d'implanter un programme en langage **Java** permettant de manipuler les codes de contrôle d'erreurs **CRC** (Code Cyclique de Redondance).

Le programme doit offrir deux fonctionnalités majeures :

1. Le calcul du code CRC à partir d'un message et d'un polynôme générateur, avec affichage du mot complet résultant.
2. La vérification de l'intégrité d'une trame reçue pour détecter d'éventuelles erreurs de transmission.

Une contrainte importante est la présentation détaillée des étapes de calcul (division binaire XOR) pour permettre à l'utilisateur de comprendre le processus.

# 2 Analyse du sujet

Le principe du CRC repose sur la division polynomiale dans l'arithmétique modulo 2.

## 2.1 Principes mathématiques

Soit $M(x)$ le polynôme représentant le message de données et $G(x)$ le polynôme générateur de degré $n$.

— On multiplie $M(x)$ par $x^n$ (ajout de $n$ zéros à la fin du message binaire).
— On effectue la division de ce résultat par $G(x)$.
— Le reste de cette division, noté $R(x)$, constitue le code CRC.

## 2.2 La division XOR

Contrairement à la division décimale, la division binaire utilisée ici n'implique pas de retenue. L'opération de soustraction est remplacée par un **OU Exclusif (XOR)**. Chaque étape consiste à aligner le générateur sous le dividende dès qu'un bit '1' est rencontré.

# 3 Choix techniques effectués

Pour répondre aux besoins de l'énoncé, nous avons opté pour une architecture modulaire.

## 3.1 Représentation des données

Nous avons choisi de manipuler les suites de bits sous forme de `String` et de tableaux de caractères (`char[]`). Ce choix est motivé par :

— **La flexibilité :** Pas de limite de taille contrairement aux types `int` ou `long`.
— **L'affichage :** Facilité pour manipuler et afficher chaque étape de la division avec des décalages visuels.

### 3.2 Structure logicielle

— **Classe `CRCManager` :** Regroupe les méthodes statiques de calcul (`calculateCRC`) et de vérification (`verifyCRC`). Elle contient également la logique de division XOR.
— **Classe `Main` :** Gère l'interface console, la saisie utilisateur via la classe `Scanner` et le menu interactif.

## 4 Résultats et tests

Les tests ont été effectués avec le polynôme générateur classique 1011 (degré 3).

### 4.1 Calcul du CRC

Pour un message 1101 et un générateur 1011 :
— Le programme ajoute trois zéros (1101000).
— Après division XOR, le reste obtenu est 001.
— Le mot complet généré est 1101001.

```
=== Gestionnaire de Code CRC ===
1. Calculer un CRC (Générer une trame)
2. Vérifier une trame (Détecter des erreurs)
3. Quitter
Choix : 1
Saisissez la suite de bits : 1101
Saisissez le polynôme générateur (ex: 1011) : 1011


--- Étapes du calcul du CRC ---
Message original : 1101
Message augmenté : 1101000
Générateur : 1011
Début de la division :
1101000
 ----
0011100
  1011 (XOR)
  ----
0001010
   1011 (XOR)
   ----
0000001
Reste calculé (CRC) : 001
Mot complet à envoyer : 1101001
```

FIGURE 1 – Capture d'écran de l'exécution du programme CRC

## 4.2 Vérification

— **Sans erreur :** La saisie de 1101001 produit un reste de 000 → "Aucune erreur détectée".

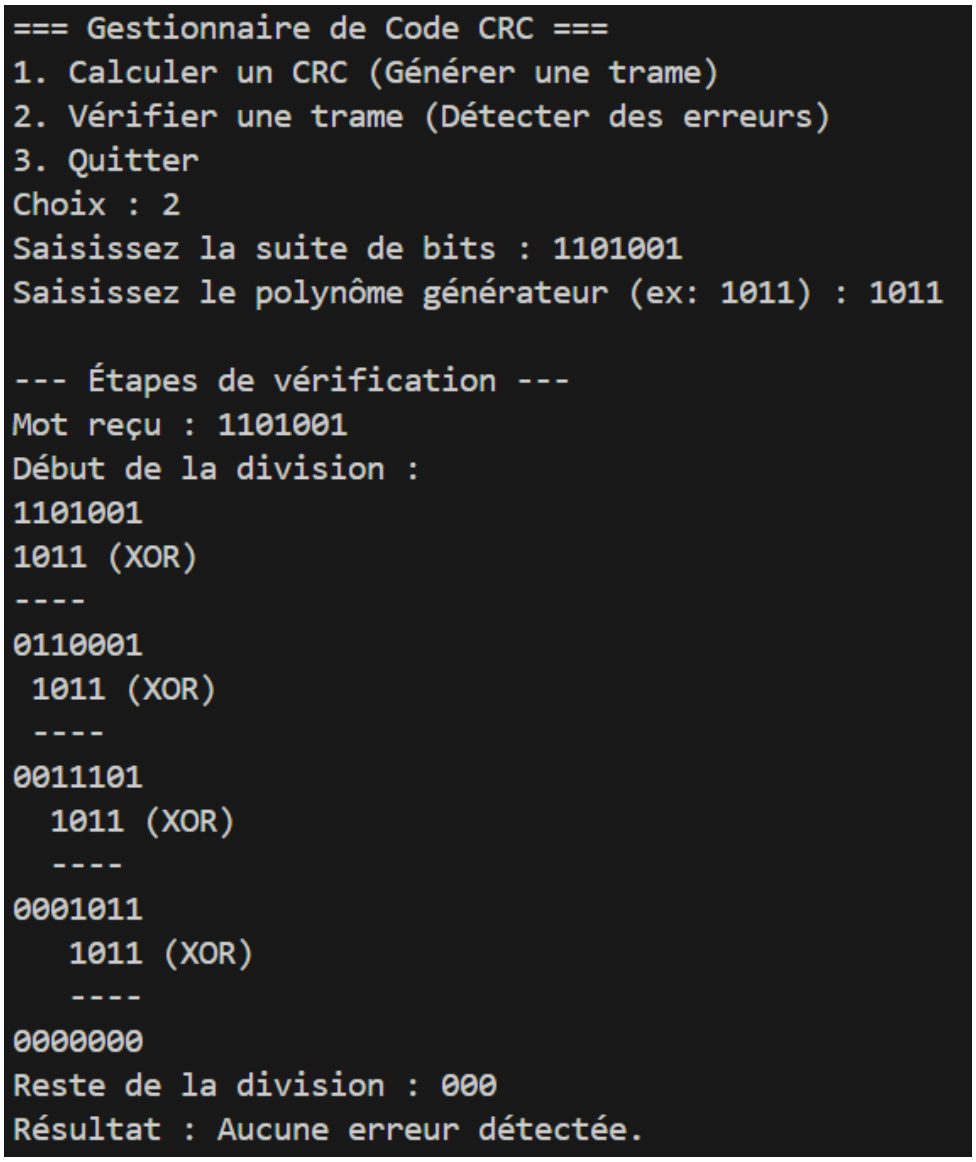


FIGURE 2 – Capture d'écran de l'exécution du programme CRC sans aucune erreur détectée

— **Avec erreur :** La saisie de 1101011 produit un reste non nul → "Erreur détectée".

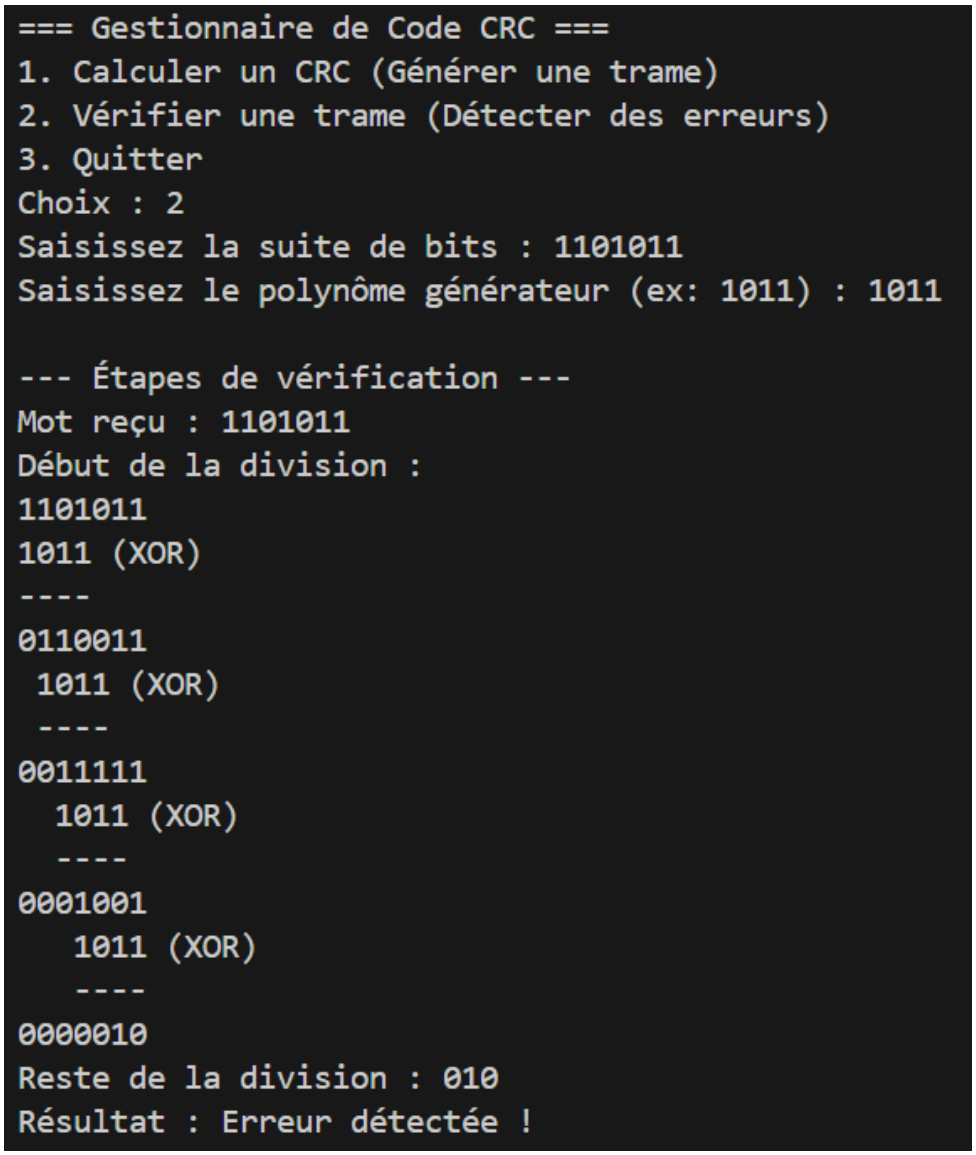


FIGURE 3 – Capture d'écran de l'exécution de la vérification du code CRC avec une erreur détectée

# 5 Conclusion

## 5.1 Difficultés rencontrées

La principale difficulté a résidé dans la gestion des zéros non significatifs. Le reste d'une division peut être plus court que le degré $n$ du polynôme. Il a fallu implémenter une méthode de formatage (`formatRemainder`) pour garantir que le CRC possède toujours la longueur attendue.

## 5.2 Limites et perspectives

Le programme actuel est robuste pour des saisies binaires, mais ne gère pas les erreurs de saisie (caractères autres que 0 ou 1). Une évolution possible serait l'ajout d'une interface graphique permettant de visualiser la division de manière plus ergonomique, ou encore l'implémentation des standards industriels comme le CRC-32.