Here is Verhoeff’s scheme using the dihedral group using special permutations. Notice that all adjacent transpositions are detected.

**Java class: ErrorDetection**

```java
// ErrorDetection.java: base class for single-digit error detection
public class ErrorDetection {
    public static void printArray(int[] a) {
        for (int i = 0; i < a.length; i++) {
            if (a[i] == 10)
                System.out.print("X ");
            else
                System.out.print(a[i]);
            if (i%5 == 0) System.out.print(" ");
        }
        System.out.println();
    }

    public static void printUnchecked(int[] a) {
        System.out.print("? ");
        for (int i = 1; i < a.length; i++) {
            System.out.print(a[i]);
            if (i%5 == 0) System.out.print(" ");
        }
        System.out.println();
    }
}
```

**Java class: VerhoeffErrorDetection**

```java
// VerhoeffErrorDetection.java: Verhoeff’s decimal error detection
public class VerhoeffErrorDetection extends ErrorDetection {
    private static int[][] op = {
        {0, 1, 2, 3, 4, 5, 6, 7, 8, 9},
        {1, 2, 3, 4, 5, 6, 7, 8, 9, 0},
        {2, 3, 4, 5, 6, 7, 8, 9, 0, 1},
        {3, 4, 5, 6, 7, 8, 9, 0, 1, 2},
        {4, 5, 6, 7, 8, 9, 0, 1, 2, 3},
        {5, 6, 7, 8, 9, 0, 1, 2, 3, 4},
        {6, 7, 8, 9, 0, 1, 2, 3, 4, 5},
        {7, 8, 9, 0, 1, 2, 3, 4, 5, 6},
        {8, 9, 0, 1, 2, 3, 4, 5, 6, 7},
        {9, 0, 1, 2, 3, 4, 5, 6, 7, 8}
    };
    private static int[] inv = {0, 4, 3, 2, 1, 5, 6, 7, 8, 9};
    private static int[][] F = new int[8][];
```
public VerhoeffErrorDetection() { // identity and "magic" perms
    F[0] = new int[]{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; // identity
    F[1] = new int[]{1, 5, 7, 6, 2, 8, 3, 0, 9, 4}; // "magic"
    for (int i = 2; i < 8; i++) {
        F[i] = new int[10];
        for (int j = 0; j < 10; j++)
            F[i][j] = F[i-1][F[1][j]];
    }
}

public static int insertCheck(int[] a) {
    int check = 0;
    for (int i = 1; i < a.length; i++)
        check = op[check][F[i % 8][a[i]] ];
    a[0] = inv[check];
    return a[0];
}

public static boolean doCheck(int[] a) {
    int check = 0;
    for (int i = 0; i < a.length; i++)
        check = op[check][F[i % 8][a[i]] ];
    if (check != 0) return false;
    else return true;
}

// main function
public static void main (String[] args) {
    VerhoeffErrorDetection v = new VerhoeffErrorDetection();
    int[] a = new int[15];
    boolean checkFlag = false;
    for (int i = 1; i < a.length; i++)
        a[i] = (int)(Math.random() * 10.0);
    VerhoeffErrorDetection.printUnchecked(a);
    VerhoeffErrorDetection.insertCheck(a);
    VerhoeffErrorDetection.printArray(a);
    System.out.println(VerhoeffErrorDetection .doChe ck(a)) ;
    VerhoeffErrorDetection.printArray(a);
    System.out.println(VerhoeffErrorDetection.doCheck(a));
    // test all adjacent transpositions
    System.out.println("\nTest all adjacent transpositions");
    for (int p1 = 0; p1 < 10; p1++)
        for (int p2 = 0; p2 < 10; p2++) {
            if (p1 != p2) {
                a[8] = p1; a[9] = p2;
                VerhoeffErrorDetection.insertCheck(a);
                // interchange
                a[8] ^= a[9];
                a[9] ^= a[8];
            }
        }
a[8] ^= a[9];
if (VerhoeffErrorDetection.doCheck(a)) {
    System.out.println("Warning: Interchange of "+
p1 + " and " + p2 + " not detected");
    checkFlag = true;
}
}
if (checkFlag)
    System.out.println("At least one transposition undetected");
else
    System.out.println("All transpositions detected");
} // end of main

Here is the output, showing a simple test, and a test of all adjacent interchanges. All interchange errors are detected.

? 75787 12372 9429
1 75787 12372 9429
true
1 75797 12372 9429
false

Test all adjacent transpositions
All transpositions detected
Here is a Java program to create standard cryptograms, as they are found in newspapers. The program will read the quotation to be scrambled into a cryptogram from the standard input. In Unix this file can just be directed into the program, as shown in the commands below. Each time it is executed, the program will create a new and unique translation table to create the cryptogram. The resulting table and cryptogram itself are output on the standard output file, which might be redirected into a named file.

```java
import java.io.*;
public class Cryptogram {

    private char[] alf = new char[26]; // translation vector

    public Cryptogram() {
        for (int i = 0; i < alf.length; i++) alf[i] = (char)('A' + i);
        randomize();
    }

    private int rand(int r, int s) { // r <= rand <= s
        return (int)((s - r + 1)*Math.random() + r);
    }

    private void randomize() {
        for (int i = 0; i < alf.length - 1; i++) {
            // Note: for a random permutation, replace "i+1" by "i" below
            // However, we want no letter to remain in its original spot
            int ind = rand(i+1, alf.length - 1);
            char t = alf[i];
            alf[i] = alf[ind];
            alf[ind] = t;
        }
    }

    public void printArray() {
        System.out.print("Alphabet: ");
        for (int i = 0; i < alf.length; i++)
            System.out.print((char)('A' + i));
        System.out.println();
        System.out.print("Translated to: ");
        for (int i = 0; i < alf.length; i++)
            System.out.print(alf[i]);
        System.out.println();
    }

    // getNextChar: fetch next char.
    public char getNextChar() {
        char ch = ' '; // = ' ' to keep compiler happy
        try {
            ch = (char)System.in.read();
        }
    }
}
```

---

**Program III.9.a**

**Cryptogram Program**

Referred to from page 58.
Here is a run of the program, first showing the quotation to be translated, and then the translated version, that is, the cryptogram:

```bash
% cat quote.text
AND WE ARE HERE AS ON A DARKLING PLAIN
SWEPT WITH CONFUSED ALARMS OF STRUGGLE AND FLIGHT,
WHERE IGNORANT ARMIES CLASH BY NIGHT.
DOVER BEACH, MATHEW ARNOLD
% java Cryptogram < quote.text
Translated to: ZUWYMPILBDJRVFHQSGAXNCTKOE
```

Now suppose one wants to have nothing but the letters in the cryptogram (no spaces, newlines, or other punctuation). This is the same as the other program, except that the `createCryptogram()` has become:

```java
public void createCryptogram() {
    char ch;
    while ((byte)(ch = getNextChar()) != -1) {
        if (Character.isUpperCase(ch))
            ch = alf[ch - 'A'];
        System.out.print(ch);
    }
}
```

Here is the output of this program:

```bash
% java Cryptogram2 < quote.text
```
Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Translated to: OXKQFDZGACIVLHEJSMBWPNURTY

OHQVFOMFGFMFOBEHOQCMIVAHZJVOAHBUFJWUAWGKEHDPBFQOVMLEDBWMPZVFOHQDVAZWUGFMFAZHEMOLHMOMLAFKVOBGXTHAZGWQENFMOXMFOKGLOWGFOUOMHEVQ

Here is what the message looks like after decrypting:

ANDWEAREHEREASONADARKLINGPLAINSWEPTWITHCONFUSEDALARMSOFSTRUGGLE
ANDFLIGHTWHEREIGNORANTARMIESCLASHBYNIGHTDOVERBEACHMATHEWARNOLD