Heilmann G4

Hugo

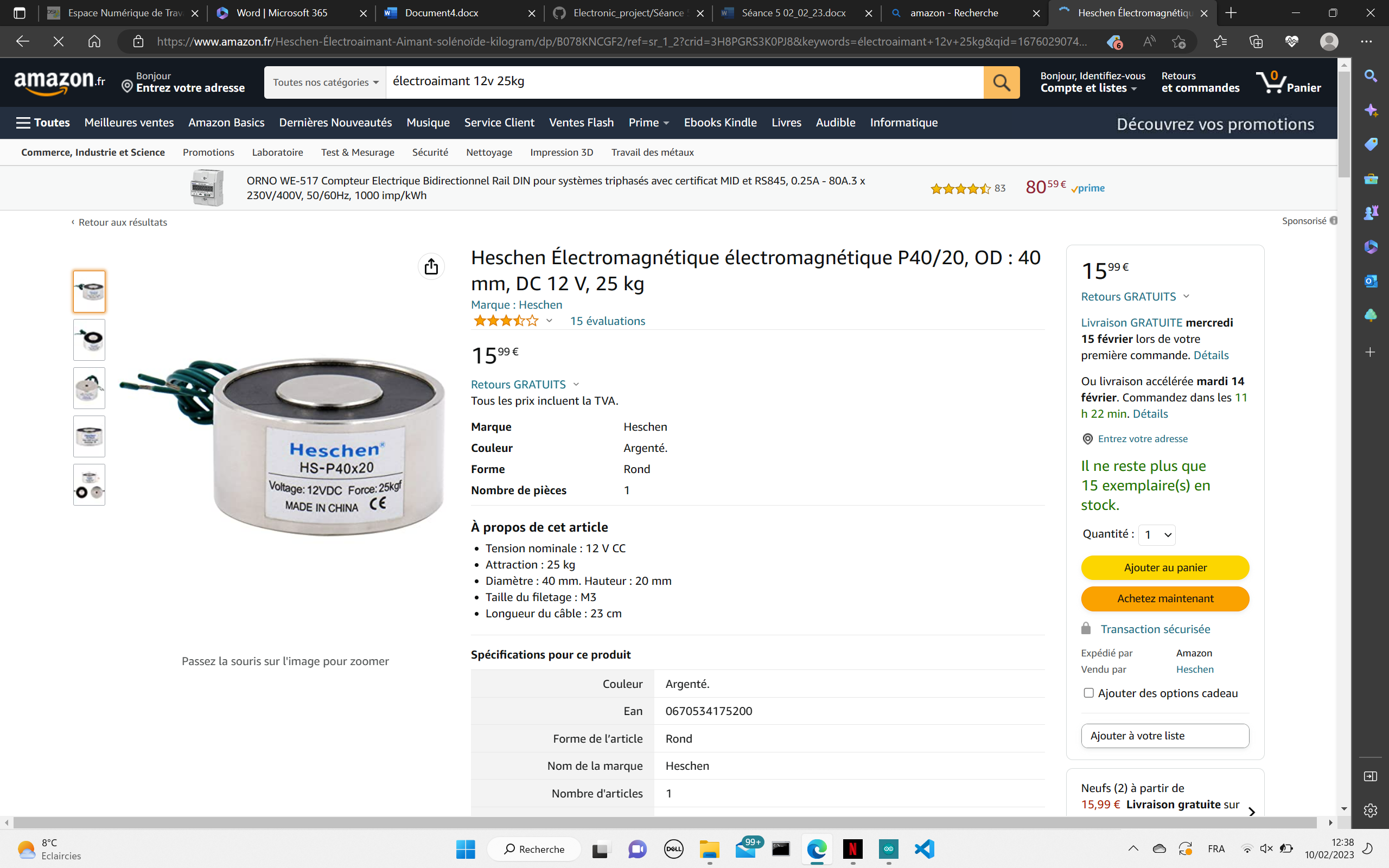
Compte-rendu séance du 09/02/2023

Lors de cette séance, nous avons, avec Camilia, rencontré un problème majeur. En effet, les pièces d’échec ont été percé et les aimants étaient maintenant capable de rentrer dans le trou situé sous chacune des pièces. Cependant, ces aimants sont très puissants et cela a eu comme conséquence que lors du déplacement d’une pièce, celle-ci attire vers elle les autres pièces présentes autour.

Nous avons donc décidé de remplacer les aimants par du fer doux tel que des vis afin que les pièces puissent être attiré vers l’électroaimant sans s’attirer entre elles.

Malheureusement, un second problème est intervenu, l’électroaimant n’est pas du tout assez puissant pour capter les vis au travers du plateau de jeu.

Nous avons donc été contraint de commander un nouvel électroaimant plus puissant. Celui-ci a une force d’attraction de 25kg comparé à l’ancien qui a une force d’attraction de 8kg ; ce qui est trois fois plus puissant.



Suite à cela, j’ai continué l’écriture du code car celui-ci est loin d’être au point.

Le tableau d’entier qui a été créé a été changé en un tableau de type “Pièces”. Ainsi, j’ai dû écrire l’intégralité des classes en C++, une pour chaque type de pièce et deux classes supplémentaires pour représenter les cases vides et mortes comme des pièces.

Voici un exemple de fichier .h :

#ifndef PIECES\_H

#define PIECES\_H

public class Pieces{

public:

Pieces(String target\_line, int target\_column, bool isWhite){

\_target\_line = target\_line;

\_target\_column = target\_column;

\_isWhite = isWhite;

}

int Transcribe(String s);

//bool legality(String wanted\_line, int wanted\_column);

virtual String type();

private:

string \_target\_line;

int \_target\_column;

bool \_isWhite;

};

#endif

Et en voici un de fichier .cpp :

#include "Cavalier.h”

int Cavalier::Transcribe(String s){

if (s=="O"){

return 0;

}

if (s=="A"){

return 1;

}

if (s=="B"){

return 2;

}

if (s=="C"){

return 3;

}

if (s=="D"){

return 4;

}

if (s=="E"){

return 5;

}

if (s=="F"){

return 6;

}

if (s=="G"){

return 7;

}

if (s=="H"){

return 8;

}

return 9;

}

bool Cavalier::legality(String wanted\_line, int wanted\_column){

if ((target\_line==wanted\_line) && (target\_column==wanted\_column)) {

return false;

}

if (type(jeu[Transcribe(wanted\_line)][wanted\_column])==Mort) {

return false;

}

if (jeu[Transcribe(wanted\_line)][wanted\_column].isWhite == jeu[Transcribe(target\_line)][target\_column].isWhite) {

return false;

}

if ((Transcribe(wanted\_line)-Transcribe(target\_line))>0) {

if ((wanted\_column-target\_column)>0) {

if (((Transcribe(wanted\_line)-Transcribe(target\_line)) + (wanted\_column-target\_column)) != 3) {

return false;

} else {

if (((Transcribe(wanted\_line)-Transcribe(target\_line))==3) || ((wanted\_column-target\_column)==3)) {

return false;

}

}

}

}

if ((Transcribe(wanted\_line)-Transcribe(target\_line))>0) {

if ((wanted\_column-target\_column)<0) {

if (((Transcribe(wanted\_line)-Transcribe(target\_line)) - (wanted\_column-target\_column)) != 3) {

return false;

} else {

if (((Transcribe(wanted\_line)-Transcribe(target\_line))==3) || ((wanted\_column-target\_column)==-3)) {

return false;

}

}

}

}

if ((Transcribe(wanted\_line)-Transcribe(target\_line))<0) {

if ((wanted\_column-target\_column)>0) {

if ((-(Transcribe(wanted\_line)-Transcribe(target\_line)) + (wanted\_column-target\_column)) != 3) {

return false;

} else {

if (((Transcribe(wanted\_line)-Transcribe(target\_line))==-3) || ((wanted\_column-target\_column)==3)) {

return false;

}

}

}

}

if ((Transcribe(wanted\_line)-Transcribe(target\_line))<0) {

if ((wanted\_column-target\_column)<0) {

if (((Transcribe(wanted\_line)-Transcribe(target\_line)) + (wanted\_column-target\_column)) != -3) {

return false;

} else {

if (((Transcribe(wanted\_line)-Transcribe(target\_line))==-3) || ((wanted\_column-target\_column)==-3)) {

return false;

}

}

}

}

return true;

}

String Cavalier::type(){

return "Cavalier";

}

Un des problèmes à régler en ce qui concerne le code est que le tableau est de type “Pièces”, or la classe “Pièces” ne possède pas de méthode “legality” (permettant de savoir si le coup demandé est légal) car cette méthode dépend du type de pièce sur lequel on applique la méthode.

De plus, je ne peux pas passer la classe “Pièces” en classe abstraite car le langage C++ n’accepte pas la création d’un tableau de type d’une classe abstraite.