introduction à la programmation

Brunner Loïc

February 10, 2018

1 introduction

- moodle: http://moodle.epfl.ch/course/view.php?id=5971
- MOOC: cours en ligne (support vidéo, en plus, on peut y accéder via le moodle je crois, seulement sur les 5 premières semaines, pas obligatoire)
- livre de Walter Savitch Absolute Java, 5th edition

2 environnement de travail

Nous allons bosser sous linux. Savoir se servir d'un terminal et se débrouiller dans les répertoires.

la commande chmod commande chmod: (u/g/o/a) +- (r/w/x), comme ça tu sais ;)

outil de développement intégré eclipse

3 programmation

3.1 a ne pas faire

Il faut toujours attiquer une tache de programmation par de la réflexion. On ne commence jamais sans prendre la peine de réfléchir.

- conception
- réalisation

3.2 algoritme

Ensemble de régles qui gèrent une suite de taitement fini. Il faut bien le conceptualiser.

4 développement de programme

4.1 language de programmaiton

Les instructions de l'ordinateur.

1. ordinateur
2. microprocesseur
3. émoire centrale
4. périférique

C'est le micro processeur qui est capable d'intermpréter l'information injectée par un programme. Les instructions sont transcrites en language machine, mais on peut aussi donner des instructin directement en langauge assembleur. Le language machine c'est le binaire.

language de programmation Des l'instant ou un dispose d'un traducteur qui permet de passer d'un language plus évolué en language assembleur.
deux types de traducteurs:

compilateur:
: traduit un programme en fichier binaire spécifique à une architechture matérielle

interpreteur:
: exécute un programme écrit dans un language de programmation sans étapes intermédaires, plus lent que compilateur

A mon grand damn, java utilise les deux

Java est capable de produire des optimisation ce qui permet d'obtenir des programmes plus rapide que s'ils étaient traduits par un interpreteur.

la JVM N'est pas dédie à interpreter du bytecode créé par java mais par du bytecode créé par tout les languages de programmations. Par contre certains languages n'utilise pas de bytecode et passe directement de la compilation à l'execution.

JIT (just-in-time) Accélère l'execution des programmes. Des portions de codes souvent utilisées sont interprétée qu'une fois puis sont ensuite directement exécutées

bytecode:
: pas vraiment lisible de l'être humain. Il est indépendant de la plateforme, on peut le lancer sur un autre processeur. N'a pas besoin du fichier source pour exécuter le programme.Seulement l'interprétation est necessaire et non la compilation.

Nous notions que java compile en bytecode et interprete par après (programme mulitplattforme)
en ligne de commande

- javac: compilation
- java: execution(tu le connais bien)

programmer c'est:

1. réfléchir eu problème
2. traduire cette réflexion en un texte, java
3. traduire ce texte en language machine
4. exécuter le programme

en pratique:

- erreurs de syntaxe
- erreurs de compilation

phase de test hyper important pour le développement, permet d'éviter des erreurs.

4.2 cycle de développement pour java

4.2.1 le language java

C'est un language orienté objet, qui est très typé.Indépentânt de la plateforme, typage fort, permet au compilateur la vérification sur la correstion de programme. Permet des projets plus ambitieux avec des connaissances moindres. Des aspect très simple dans d'autres languages ne sont pas montré par java, certains concepts de base sont cachés.

main programme principal, méthode qui est exécutée au début du programme. Une classe est une brique de base, elle contient un certain nombre d'instructions.

phrase utile pour la compréhansion les structures de données organisent les traitements qui opèrent sur les données

4.2.2 variable

a besoin d'un type et de son identificateur. Valeur définie la première fois lors de l'initialisation. on écrit:
type id; = valeur; les différents types élémentaire:

- int
- double
- char
- boolean

identificateur c'est simplement une séquence de caractère qui permet d'identifier et d'utiliser la variable (son nom quoi)

déclaration:
donner un nom à la variable

initialisation:
donner la première valeur à une variable

On peut déclarer plusieurs variables par ligne mais il est conseillé de ne pas le faire pour la clarté de la lecture du programme. Une variable est une zone de mémoire. Elle commence par une minuscule, conventions(lire les conventions de style)

int Permert de représenter des entiers. Un entier est codifié sur 32 bits(maximum de 4 octets). D'autres types pour les eniter:byte(1 octet),short(2octet),long(8octet)

double maximum 8 octet: 64 bits. On met un point et pas une virgule et on note e au lieu de 10 en notaiton scientifique.autre variable float(4 octets)

constante mettre final avant la variable. Très important quand c'est possible, car garde fous.

affectation On évalue la partie a droite pour l'affecter a gauche: x = x + 1;

final:
permet de créer des constante nom modifiables après leur première initialisation

affectation:
donner une valeur à une variable

resumé les différents types de variables lpermettent la vérifiaction de la coérance du programme et une utilisation éfficace de la mémoire.

transtypage on peut passer un int en double. Pour faire l'inverse il faut un (int)devant la donnée. Mais a utiliser qu'en ca sde dernier recours. Peut mettre en évidence un soucis sur la perception de la conception du problème. IL faut que les intentions de programmeur soient claires. De plus, le transtipage induit une perte de donnée.

valeur tronquée on garde juste la partie entière du double par exemple(pas la même chose qu'un arrondi)

notations abrégées ++x x++ et –y et y– la différence: ne change rien si utilisé tout seul, mais a une incidence si on commence a mettre cette instruction dans une expression. le ++x permet d'incrémenter d'abords et de faire le calcul après alors que x++ permet de faire le calcul avec l'objet avant son incrémentation.

4.2.3 opération

Il est conseillé de les parentaiser.

ordre en cas de non parenthaisage:*/%+-

4.2.4 les classes

Les noms de classes commencent toujours par une majuscule et chaque mot commence par une majuscule. Essayer de rester clair dans les noms de class et de variables.

Scanner une variable Scanner se note: Scanner a = new Scanner (system.in); la méthode retourne un string et permet de lire différentes variable, ligne, données,... entrées dans la console. elle retourne un int , double ou string en fonctin de la fonction

4.2.5 commenter

Commenter de facon utile et intelligente est un devoir pour le programmeur.

5 les structures de contrôle

bloc séquence d'instruction définies entre { et }

5.1 condtions

chaque if est soumis a des expressions de condition. Elles sont exprimées au travers des opérateur logique. ((non)égalité, plus grand et plus petit ou égal.) Normalement tu devrais pas avoir trop de problèmes pour faire ca. réfère toi au site du zéro pour connaître tous les opérateurs logiques. Quand java évalue que ce qui est trictement nessessaire alors lors d'une opération && si le premier est faux, il n'évalue que le premier.

branchement conditionel:
permet d'éffectuer des instruction selon certaines conditions

5.2 while

connaître la différence entre un while et un do ... while

opérateur ternaire ? nous pouvons exprimer comme suit; (expression logique) ? ExpressionVrai : ExpressionFaux
retourne ExpressionVrai si l'expression logique est vraie sinon ExpressionFaux

5.3 variables déclarées en dehors de la méthode main

elle sont apellées variables globales

5.4 switch

permet de tester une valeur entier avec le switch(i)case 1:...break; case 2...default:instruction; On peut mettre un switch sans mettre de break si on ne met pas de break toutes les instructions rencontréees par la suite sous les case. Permet de remplcer des instructions if plus lourdes. Très utilisé pour manipuler des types ennumérés... Attention les case doivent être des constantes.

attention les case doivent être des constantes

5.5 les itérations

on utilie le type for quand le nombre d'itération est connu.for(int i=;1<10; ++i) Il est possible de déclarer plusieurs variables dans l'instruction for(int i=0,j=0;...;...)

5.6 break et continue

break:
: on sort tout de suite de la boucle sans regarder la condition

continue:
: on retourne directement vérifier la condition et on refais la boucle si condition vraie

5.7 tableaux

Avec les tableau ou les chaines de carartère, ils contiennent des références vers les différentes variables du tableau. iILs peuvent etre dynamiques ou bine statiques. le tableau est une structure de donnée:

- regroupe n valeurs du même type
- donne le même nom aux n valeurs
- ennumère les valeur de 0 à n-1
- les n valeurs sont les éléments du tableau

la gestion des collections Nous pouvons gérer les collections comme suit:for (type variable : collection)
en gros, le tableau est une collection de type qui vont être entrés dans variable les uns après les autre avant de chaque fois faire une boucle

5.7.1 les String

nous ne pouvaons pas utiliser le == mais utiliser .equals() (sinon on compare les références)
un littéral qui est entré par un utilisateur n'est pas dans le pool des littéraux, pour faire des comparaisons, il faut mettre reponse=reponse.intern();

5.7.2 les ArrayList

ils doivent toujours être initialisé comme contenant un type évolué
Donc attention, les éléments d'un tabeau dynamique sont toujours des références

5.8 les méthodes et la réutilisabilité

une fonction contient toujours:

un nom:
: référence à l'objet methode quand il est invoqué

arguments:
: les entrées, ensembles de réflérence à des objets définis à l'extérieur de la fonction

un corps:
: code à réutiliser, ce qui justifie l'existance de la fonction

variables internes:
: variables n'existant qu'à l'intérieur de la fonction

une valeur de retour:
: valeur que retourne la fonction à l afin des opérations

surcharge des méthodes Deux méthodes peuvent avoir le même nom si elles n'ont pas la même liste d'arguments

6 programmtion orientée objet

l'orineté objet permet d'aporter un lien sémentique et logique entre les éléments des programmes

6.1 abstraction

partir de qqch de spécifique, et arriver à une idées plus abstraite.

encaplsuler:
: tout mettre dans une boite, regrouper un ensemble de données et de méthodes

abstraire:
: partir de qqch de très concret et en resortir un concept (une largeur hauteur= un rectengle)

attribut:
: données inclues dans un objet

le méthode:
: le traitements d'un objet

le corps d'un objet:
: regroupe toutes les méthodes/attributs qui ne sont accéssible qu'à l'intérieur de l'objet

interface utilisateur:
méthodes/attributs qui sont accessibles au monde extérieur

encalsulation permet d'abstraire qqch de spécifique pourarriver à qqch de plus générique, mettre ensembles des éléments qui caractérise l'objet, mais aussi, offrir des fonctionalités au monde extérieur. Ainsi, il existe des chose ilnutilisables par le monde extérieur

classe:
résultat du processus d'abstraction

instance:
réalisation particulière d'une classe

variables d'instance:
les attributs d'une classe

si le programmeur modifie sa classe, le monde extérieur doit pouvoir continuer à tourner avec le même outillage

attention les attributs d'une classe ne doivent pas être accessibles de l'extérieur, sinon on perd l'aventage de l'objet. Ainsi, le monde extérieur n'a de vision de l'objet que ce que le programmeur concepteur lui permet de voir.

6.2 type de données

Quand tu défini une classe, tu défini un nouveau type de donnée. Désormais, tu peut déclarer une variable(instance) qui as toutes les caractéristiques de l'objet.

une instance:
une réalisation particulière d'un objet

6.3 class

- mot clé: class Rectangle{}

on peut le faire tout dans un même fichie, mais si les classes occupent des problèmes centraux distincts, on les déclarent dans des fichiers différents

la seule classe exécutable est le main

- attributs: variables d'instance: type nom_attribut;(on le fait usuellement en début de classe)
- pour accéder à un attibut: instance.nom_attribut;
- les métode de classe ne doivent pas contenir le mot clé static
- nous n'avons pas besoin de passer les attibut de la classe aux fonctions de la méthode, il ssont directement accessibles

6.4 niveau d'accès

définition de l'interface utilisateur:permet de choisir ce que peut toucher l'extérieur et ce qu'il ne peut pas.

private:
: ne permet pas l'accès au monde extérieur

public:
: offrir le service à l'extérieur

6.5 instance et valeurs spécifiques

masquage:
: quand le nom d'un paramêtre est le même que celui d'un attibut(utiliser this pou ry remédier, le reste du temps this est pas obligatoire)

Nous voulons maintenant spécifier certaines valeurs durant l'instanciation de la classe

accesseur(get):
: permet d'accéder aux attributs de la class en les laissant privés

manipulateur(set):
: petmet de modifier les attibuts de la classe sans mettre les attributs en public

si nous les mettions tous en public, les personnes de l'extérieur peuvent les déclarer n'importe comment. Ainsi, nous pouvons encadrer l'utilisation de l'objet

attention il ne faut pas tout mettre en getter et setter! il faut savoir gérer ce qui est utile et ce qui ne l'est pas. De plus, il ne faut pas casser l'encapsulation.

6.6 initialisation

se note NomClass(liste des arguments){
initialisation des attibuts
}

Tout comme les fonction, nous pouvons surcharger les classe tout comme les fonction

constructeur:
: fonction, sans valeur de retour, qui permet d'instancier une classe(le constructeur a le même nom que la classe)

constructeur de copie:
: constructeur qui fait une copie de tous les attributs terme à terme (évite le passage par référence)

Nous notons que le constructeur par défaut initialise tous les attributs avec leur valeurs par défaut

instanciation: Rectangle salut = new Rectangle(a,b);

6.7 construction

si aucun constructeur n'est spécifié, java met en place un constructeur par défaut. Il initailise toutes les variables à 0. Par contre le constructeur par défaut disparaît dès l'apparition d'un quelconque constructeur

mise en place les constructeurs doivent être notés dans la classe comme des fonction avec le nom de la classe.
attention, il est recommendé de toujours mettre un constructeur par défaurt même vide, dès que le constructeur est défini. Comme ca il n'existe plus du tout et on ne peut plus exploiter les valeurs par défaut.

Ne pouvons aussi initialiser un objet par copie: public UneClasse(UneClasse obj)
Dans ce genre d'instanciation, on peut accéder aux attributs de l'autre classe même les private car nous somme en fait dans le même classe. Ce que nous empêche de créer plein d'accésseurs.

6.8 portée de la classe /instance

en java, une classe a accès aux membres de toutes ses propres instance!!! insi nous pouvons travailler avec plusieur rectangles dans le même

6.9 petite paraentaise sur la mémoire

le Garbage collection, ramasse miettes, si une instance n'existe plus elle est détruite et les espaces mémoires se déréservent, pas comme en c++

6.10 référence

tous les objet sont manipulés par référence, attention, cela doit être pris en compte et il faut être attentif aux effets de bord.

null:
signifie que la variable ne référence vers aucun objet

toString:
permet de gérer l'affichage d'un objet, prévu par java, il suffit de la créer pour que ca marche

equals:
pareil, suffit de définir, et on peut mettre un =

les = et les comparaison se font rapport aux référence en mémoire

Pour comparer les objet il est conseillé de créer une methode équals qui renvoie un boolean selon les critères de comparaisons qui semblent pertinents

6.11 héritage

on utilise le mot clé extends, et on crée des sous classes et des super classes.

hiérarchie de classe:
: réseau de dépendances dont chaque noeud hérite des propriétés des noeuds précédent en remontant à la racine

attention l'héritage dit est un pas a un (un orc est un avatar, et non a une hache(pour ca il faut créer un attribut))

on appelle ca aussi un classe parent, et des classes enfants, en java nous ne pouvons avoir qu'un parent par classe
si on a besoin de plusieurs parents, on a pas le choix, il nous faut utiliser des interfaces

mot réservé protected:
: pour utiliser les attibuts de la classe parent dans la classe enfant, protected est euqivalent à private mais pour toutes les classes enfants)attention il est en fait accéssible par toutes les classe du meme package donc faire tres tres gaffe quand tu utlilise

6.11.1 constructeurs

chaque attribut doit être construit dans la classe ou il est! pas dans les sous-classes. c'est le constucteur de la super classe qui qui recoit la charge de construire les attributs des sous classes.

argument super();:
: permet d'appeler le constructeur de la super classe(super est une méthode qui initialise polus haut)

argument this();:
: comme super mais quand il y a surchage des constructeurs, permet d'appeler un constructeur de la même classe

On ne peut pas faire cohabiter super et this car les deux doivent être les prmières instructions du constructeur

si une méthode n'est plus pertinente pour une des classe enfant il suffit de redéfinir la méthode dans la classe enfant, et la définir comme on le souhaite

super tu te rends compte que super c'est juste le this pour la super classe, tu peut appeler des fonctions du dessus aver super.fonction();

masquage:
: quand le même nom de variable est utilisé sur plusieurs niveaux(shadowing),
quand le même nom est utilisé pour des méthodes a plusieurs niveau(overriding)

instanceof permet de tester le type d'un objet
b=(oz instanceof Sorcier);
Attention, cet opérteur pourrait nous amener à ne pas utiliser le polymorphisme alors qu'il faudrait

6.12 le polymorphisme

polymorphisme de traitements:
surcharge des fonctions/méthodes

polymorphisme de données:
inclusion(le meme code peut être appliqué é des types de données différents liés entre-eux par un sous-typage) et paramétrique(le même code peut être appliqué à n'imppostre quel type de données)

le code s'adapte automatiquement aux modifications d'objets, c'esst un outil très puissant

6.12.1 le polymorphisme d'inclusion

se met en oeuvre grace à l'héritage dans les hiérarchies de classe, c'est un truc que tu sais faire fais toi pas trop de soucis pour ca

6.12.2 le méthodes et classe abstraite

Quand les super classe ne font pas sens d'être utilisée, il suffit de mettre le mot clé abstract devant pour que la tentation ne vienne passe

O n peut aussi le mettre devant des méthodes, ce qui sert a définir l'existance d'une fonction tout en ne sachant pas encore ce qu'elle contient, on peut l'abstriare afin de ne pas avoir la définir dans la super classe et la redéfinir dans les sous-classes. Une méthode abstraite n'est pas completement spécifiée ce qui nous permet de la redéfinir sans problèmes. A noter que la classe qui contient une méthode abstaite doit également l'être

U ne classe abstraite n'est pas instanciable. Ce qui est logique puisque certaines méthodes ne sont pas encore définies. Nous notons qu'une classe abstraite peut hériter d'une méthode abstraite

l e mot clé abstract est une bonne méthode pour clarifier les intentions du programmeur

6.12.3 la résulution dynamique des liens

c'est le type effectif, et non le type apparent qui est pris en compte en java

6.12.4 polymorphismes

polymorphise de traitement:
c'est le mécanisme de surcharge des méthodes, on peut utiliser le même nom pour des fonctions similaires

polymorphisme de donnée:
: le même code peut être appliqué à des données de type différent, grace à la relation de sous-classe

6.12.5 la méthode equals

commencons par faire un détour par la notion de transtypage

getClass:
: est un méthode de Object qui retourne la classe d'un objet, permet de retourne false si les deux classes ne sont pas les mêmes, ce qui est logique

le getClass et plus précis, et potentiellement meilleur que le instanceOf
le java est un language à typage fort, il faut donc toujoures repecter les types de variables

o n peut convertir une super classe en sous-classe avec cette notion :(type)maVariable
(on peut faire des conversions de type vers le bas mais pour plus d'exemples voir le diapos du prof)

class Objet:
: c'est une super classe au dessus de toutes les autres

la méthode equals il est toujours conseillé de redéfinir les fonctions equals de nos classe pour chaque type. c'est meiux qu'effectuer une surcharge
pour voir des exemples d'equals dans le cours de la prof

6.12.6 le modificateur final

permet d'indiquer que des éléments du programme ne doivent pas être modifiés, on ne peut pas lui affecter une valeur plus d'une fois

p ermet aussi de clarifier les intentions du programmeur

a noter que pour un objet en final, on ne peut pas modifier la référence mais on peut modifier sans soucis l'objet référencé
Empêche une classe d'avoir un jour une sous classe, ou un e méthode d'être redéfinie dans une sous-classe

a noter si on met final devant un objet référencé, on peut modifier les attribus et accéder aux méthodes mais on ne peut pas redéfinir l'objet, ca peut avoir du sens si on est dans une classe qui a deux arguments du même non. En plus, cela permet de clarifier les intentions du programmeur, mais pour le moment ca ne te sert par à grand chose mais tu les verras plus tard-

s i dans une classe on crée une méthode final, aucune sous-classe ne pourra la redéfinir

6.13 interface

Java ne permet pas l'héritage multiple, ce qui peut induire un certain nombre de complications à l'utilisation, gestion de l'héritage de membres identiques
une interface permet d'imposer à certaines classe d'avoir un certain nombre de méthodes communes sans qu'elles héritent d'une super classe particulière

- pas de constructeur
- éventuellemetn des constantes
- éventuellement des méthodes abstraites
- on peut définir des entêtes de méthodes pour imposer leur existance
- on utilise le mot réservé: interface
- pour dire qu'une classe met en oeuvre les méthodes d'une interface: implements

A noter que le instanceOf ne marche pas pour tester si la classe a une interface, car une iterface n'est pas un classe!
On peut déclarer des variables de type interface, y affecter un objet d'une classe qui implémenter l'interface, faire un transtype vers l'interace

o n peut même avoir des liens d'héritage entre les interfaces
avec une super classe on attribue un lien est un alors qu'avec une interface on attribue un lien a un, ou se comporte comme

6.14 le modificateur static

On peut l'appliquer aux attibuts et aux méthodes.

pour une variable:
la valeur de la variable est partagée entre toutes les instances de la classe

pour une méthode:
on peut appleler la méthode sans construire l'objet

une variable static est une bonne représentation d'une valeur commune à toutes les classes, mais pas très bon au niveau de l'encapsulation.
UN attribut s'atic est accessible depuis l'extérieur de la classe: A.Attribut
un variable d'instance est spécifique à l'instance alors qu'une variable de classe est la même quelle que soit l'instanciation de la classes

attention il est important de comprendre que le mot clé static sert surtout à comprendre les intentions du programmeur, à noter que le static utilisé n'importe comment va simplement tuer l'orienté objets
a noter que comme une constante ne peut être modifiée, il fait sens qu'elle soit la même pour toutes les classes donc on met hyper mega souvent: final static

6.14.1 les méthodes static

Nous pouvons accéder aux méthodes statics sans créer d'instances de la classe: Classe.méthode, c'est ce que tu fait avec la classe Math, on peut aussi accéder aux attributs statics de cette facon

attention le this ne dois pas être utilisé dans une méthode static car la classe n'est pas forcément instanciée. De même, elle ne peut accéder à des méthodes qui accèdent au this
bref on le fait pour les classes contenant un certain nombre d'utilitaires

6.15 bon usage des références en java

attention a bien penser l'interface d'utilisation des classes.
une bonne implémentation doit protéger l'utilisateur des détails d'implémentation, et le protéger des changement qui peuvent être faits à l'intérieur du code

6.15.1 failles d'encapsulaiton (privacy leaks)

pour eviter qu'une getter retourne une référence à un attribut privé, il faut renvoyer une copie profonde de l'attribut, si c'est un objet qui fonctionne par référence.

classe mutable:
: si la classe contient des méthodes publiques permettant de modifier ses instances

classe immutable:
: n'est pas une classe mutable

constructeur de copie essenciel pour faire des copis d'objets, ces copie doivent être profondes. Des lors toutes les clsses doivent avoir des constructeurs de copie profonds.
fait gaffe, il est très facile d'avoir des failles d'encapsulaiton!!!

clone méthode héritée de Object, a le même role que le constructeur, mais agit de facon polymorphique, ce n'est pas le cas des constructeurs de copie.
on peut le faire facilement, mais il y a une méthodologie très précise pour utiliser la méthde clone();, Mais pour le moment c'est une intro, tu ne dois pas encore l'utiliser sinon ca va pas aller.

- invocation de la méthode clone des super-classes
- utilisation de l'interface clonable
- gestion des exeptions

```
public Oral clone() {
return new Oral(this);
}
```

6.16 gestion des exeptions

attention c'est important

exceptions premettent d'anticiper les erreurs. La question: qu efaire concrètement en cas d'erreur?

- trycatch() : indique un bloc réceptif aux erreur,intercepte les erreurs associées
- throw new Exception(); indique l'erreur
- finally: indique ce qu'il faut faire apès un bloc réceptif
- new exception (objet d'exception)

on lance une bouteille, et on l'attrape dans le programme principal, permet de prévoir une erreur à un endroit, et é la gérer à un tout autr enedroit, un endroit sait la gérer
quand on a une bouteille qui est lancée, mais pas ratrappée, provoque un arret du programme. Nous nottons que si l'exception peut être traitée localement, en pa sutiliser ce mécanisme.

6.16.1 throw

instruction qui signale l'erreur.
Exception est un objet qui est lancé (descends de la classe Throwable)
deux constructeurs:

1. new Exectpion();
2. new Exception(String message);

chaque sous-classe de Trowable contient des classe qui gèrent différents types d'erreurs.

6.16.2 le block finally

permet de finaliser les traitements: le but est de faire le ménage (fermer les fichier, des connexions,...), remise en état desressources
le block finally est lancé, qu'il y ai une exception ou pas

nosu pouvons être amené à relancer l'exception depuis un niveau intermédaire, pour avoir plus de précisions.

6.16.3 personaliser ses exception

dans la nature du message
mais nous pouvons aussi creer une classe d'exception plus personalisée

6.16.4 attention

le mécanisme des excpetions est couteux(bcp plus qu'un simple if then else)