



UNIVERSITÉ DU HAVRE

COMPTE RENDU DU PROJET PLURIDISCIPLINAIRE

---

## Ensoleillement

---



*Par :*  
BEN MARZOUK Mohamed  
L2 MI

*Encadrant :* M.FRÉDÉRIC SERIN

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Execution du Programme . . . . .	2
<b>2</b>	<b>Position du soleil</b>	<b>4</b>
2.1	L'angle Azimut . . . . .	4
2.2	L'angle d'élévation . . . . .	6
2.3	L'angle de déclinaison . . . . .	7
2.4	L'angle Horaire . . . . .	8
2.5	lst . . . . .	8
2.5.1	Temps locale . . . . .	9
2.5.2	Time Correction Factor . . . . .	9
2.5.3	l'équation du temps . . . . .	10
2.5.4	B (la fraction année en radians) . . . . .	10
2.6	class SolarParameters . . . . .	11
<b>3</b>	<b>La portée d'une ombre</b>	<b>12</b>
<b>4</b>	<b>Énergie reçue par une surface</b>	<b>13</b>
4.1	Conclusion . . . . .	13
<b>5</b>	<b>Resume en Anglais</b>	<b>14</b>

# 1 Introduction

Dans le cadre de notre programme en L2 MI, les étudiants sont amenés à choisir et à effectuer un projet proposé par les enseignants de l'UFR du HAVRE. J'ai choisi le projet «l'ensoleillement» encadré par M. Frédéric Serein.

Le soleil est une pure source d'énergie qui soutient la vie sur terre, sa chaleur contribue à notre climat, bouillant l'eau et produisant le vent qui fait bouger les nuages, sa lumière est aussi absorbée par les plantes pour afin de mûrir pour produire de l'oxygène qui est nécessaire pour la survie des créatures terrestres. Sans sa lumière et sa chaleur nous n'auront ni eau ni température qui nous est supportable, ce qui veut dire que sans soleil la vie sur terre disparaîtra. Certaines civilisations le vénèrent, le considérant un Dieu.

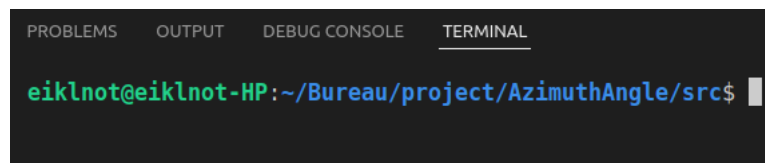
Les scientifiques ont passé des générations à faire des recherches au sujet de notre étoile, déterminant que le soleil est le centre de notre système, et que les saisons sont définies par la rotation de la terre au tour du soleil, et continuent à ce jour à l'étudier.

Le soleil est aussi un guide, que les explorateurs et marins ont appris à utiliser pour pouvoir se repérer et naviguer grâce à sa position dans le ciel.

## 1.1 Execution du Programme

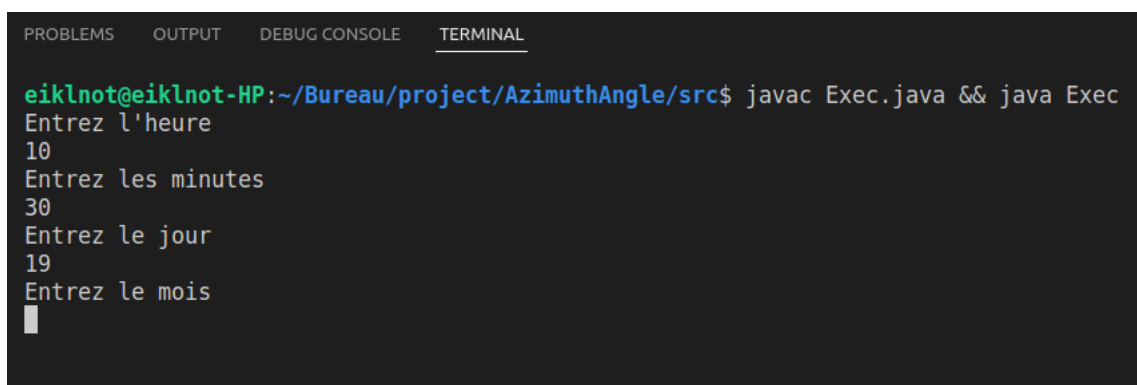
Dans ce projet j'essaierai de développer un programme en JAVA, qui nous aidera à calculer la position du soleil selon une heure, une date et des coordonnées précises. Ensuite de nous donner comment calculer la longueur d'une ombre. Puis la quantité d'ensoleillement reçue par une surface.

Pour exécuter le programme JAVA il faut aller au terminal, changer de répertoire et aller vers le répertoire **src**



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
eiklnot@eiklnot-HP:~/Bureau/project/AzimuthAngle/src$
```

Ensuite écrire et exécuter la commande suivante : `Exec.java java Exec`, et il vous sera demandé d'entrer l'heure, les minutes, le jour, le mois, l'année, la latitude, la longitude et UTC (universal time coordinated).



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
eiklnot@eiklnot-HP:~/Bureau/project/AzimuthAngle/src$ javac Exec.java && java Exec
Entrez l'heure
10
Entrez les minutes
30
Entrez le jour
19
Entrez le mois

```

Dans cet exemple nous allons prendre les paramètres suivants : 10h 30min le 21/6/2022, la latitude 49°, la longitude 2°19' (en mettant la virgule au lieu du point) et en UTC nous mettons 0 pour signifier Greenwich.

```
eiklnot@eiklnot-HP:~/Bureau$ python3 main.py
Entrez l'heure
10
Entrez les minutes
0
Entrez le jour
21
Entrez le mois
6
Entrez l'année
2022
Entrez la latitude
49
Entrez la longitude
2,19
Entrez le UTC
0
```

Ainsi nous aurons comme resultat l'angle de l'élévation, l'angle de l'azimuth la relation de l'ombre, et d'une etude energetique recomme montré dans la figure ci dessous

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

L'angle Azimuth (A) est : 140.3591752672196
*****
L'angle de l'elevation (alpha) du soleil est : 59.695731093881164
*****
L'angle de declinaison (delta) du soleil est : 23.369707423042353
*****
L'angle horaire (hour) du soleil est : -20.529191071531272
*****
la longueur de l'ombre (L)=la hauteur de l'objet (C) x cotang(l'angle de l'elevation (alpha))
*****
L=Cx0.5844527713006744
*****
S'(la pente (p),Angle azimut à la normal (An)) = derivé de l'angle incident (i')xcos(angle incident (i) )
*****
cos(i)=sin(p) x cos(alpha) x cos(A-An) + cos(p) x sin(alpha)
*****
cos(i) = sin(p) x cos(140.3591752672196 - An) +cos(p) x 0.8633579576394883
*****
i = acos(sin(p) x cos(alpha) x cos(A-An) + cos(p) x sin(alpha))
*****
i = acos(sin(p) x cos(140.3591752672196 - An) +cos(p) x 0.8633579576394883 )
*****
i' = 1/Math.sqrt(1-i^2)
*****
i' = 1/Math.sqrt(1- (acos(sin(p) x cos(140.3591752672196 - An) +cos(p) x 0.8633579576394883 ))^2)
*****
S'(p,An) =1/Math.sqrt(1- (acos(sin(p) x cos(140.3591752672196 - An) +cos(p) x 0.8633579576394883 ))^2) x (sin(p)
*****

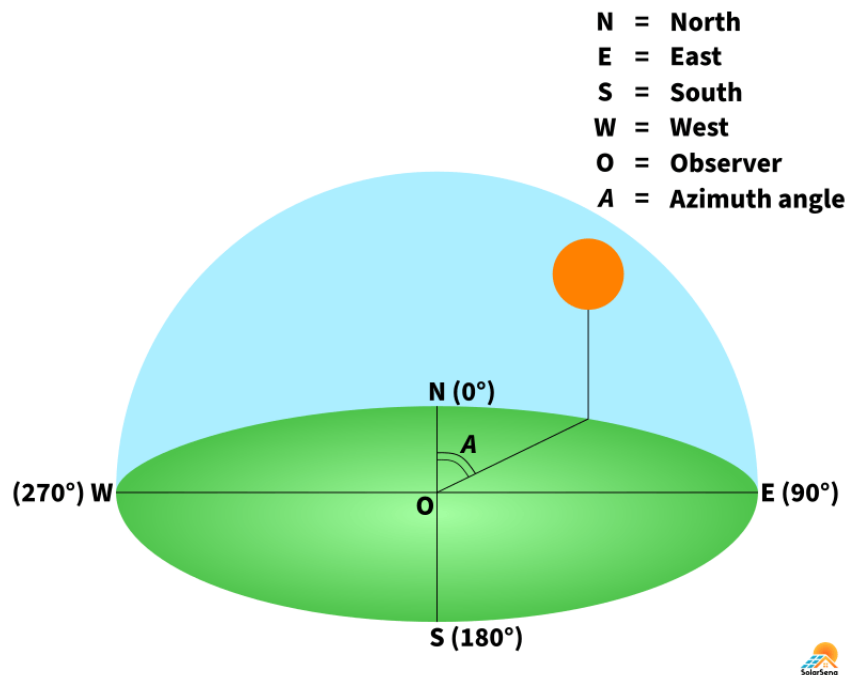
eiklnot@eiklnot-HP:~/Bureau/project/AzimuthAngle/src$
```

## 2 Position du soleil

La position du soleil est constituée principalement de l'angle de l'élévation, et de l'angle azimut.

### 2.1 L'angle Azimut

L'angle azimut définit les coordonnées horizontales du soleil par rapport au point d'observation. Il est mesuré depuis le nord à  $0^\circ$  et à  $360^\circ$ , et l'est à  $90^\circ$ , à l'ouest à  $270^\circ$  (ou  $-90^\circ$ ), le sud à  $180^\circ$ , comme montré dans le graphique ci-dessous.



Durant le lever du soleil l'angle de l'azimut est aux alentours de  $90^\circ$  vers l'est, et au coucher du soleil il est aux environs de  $270^\circ$  vers l'ouest. L'angle est positif dans le sens d'une montre.

Dans la classe `AzimuthAngle` on a comme attribut : l'angle de l'élévation, l'angle de déclinaison, l'angle horaire et la latitude.

```
private DeclinationAngle declinationAngle;  
private double latitude;  
private ElevationAngle elevationAngle;  
private HourAngle hourAngle;
```

La méthode principale de la classe `AzimuthAngle` est la méthode `get_value()` qui calcule la valeur de l'angle azimut.

```

public double get_value() {
    double delta = declinationAngle.get_value();
    double alpha = elevationAngle.get_value();
    double hour1 = hourAngle.get_value();

    if (hour1 >= 0) {
        return 360 - Math.acos((Math.sin(delta * (Math.PI / 180)) * Math.cos(latitude * (Math.PI / 180))
        - Math.cos(delta * (Math.PI / 180)) * Math.sin(latitude * (Math.PI / 180))
        * Math.cos(hour1 * (Math.PI / 180)))
        / Math.cos(alpha * (Math.PI / 180))) * 180 / Math.PI;
    } else {
        return Math.acos((Math.sin(delta * (Math.PI / 180)) * Math.cos(latitude * (Math.PI / 180))
        - Math.cos(delta * (Math.PI / 180)) * Math.sin(latitude * (Math.PI / 180))
        * Math.cos(hour1 * (Math.PI / 180)))
        / Math.cos(alpha * (Math.PI / 180))) * 180 / Math.PI;
    }
}

```

Cette méthode a deux conditions, car si L'angle Horaire (H) est supérieur ou égal à 0 alors l'équation de l'angle azimut est :

$$A = 360 - \cos^{-1}(\sin(\delta) \cos L - \frac{\cos \delta \sin L \cos H}{\cos \alpha})$$

Avec  $\alpha$  l'angle de lélévation,  $\delta$  l'angle de déclinaison, L la latitude et H l'angle horaire.

Dans le programme on multiplie ce qu'il y a à l'intérieur des cos et sin par  $\frac{\pi}{180}$ , et multiplie le résultat du  $\cos^{-1}$  par  $\frac{180}{\pi}$  afin d'avoir un résultat plus précis

Et si  $H < 0$  alors :

$$A = \cos^{-1}(\sin(\delta) \cos L - \frac{\cos \delta \sin L \cos H}{\cos \alpha})$$

On a ensuite la méthode print\_value(), qui a comme principe d'afficher le message "L'angle Azimuth (A) est : " ainsi que le résultat attendu.

```

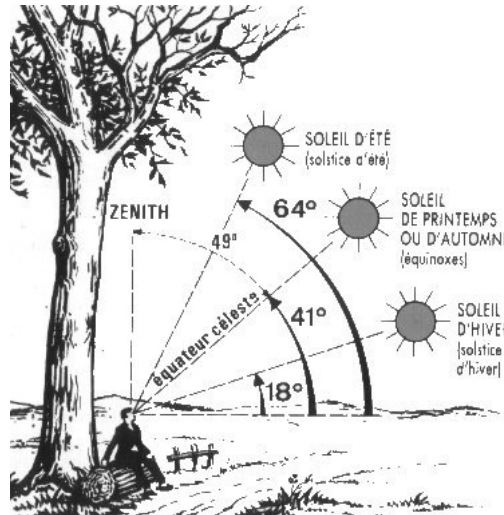
public String print_value() {
    return "L'angle de l'elevation (alpha) du soleil est : " + get_value()+"\n *****";
}

```

les étoiles ont été ajoutées pour meilleure lisibilité.

## 2.2 L'angle d'élévation

L'angle de l'élévation du soleil est la distance angulaire entre le plan horizontal du point d'observation et le soleil dans le ciel. Quand la valeur de cet angle est positif, c'est à dire que le soleil est au dessus de l'horizon ce qui signifie qu'il fait jour, alors que quand la valeur est négatif cela veut dire que le soleil est au dessous du plan horizontal, signifiant qu'il fait nuit.



Dans le lever du soleil l'angle d'élévation est proche de 0°, et est à sa valeur maximale en demi-journée. Dans notre cet code cet angle est représenté dans la class ElevationAngle, qui a comme attribut : L'angle horaire (H), la latitude et l'angle de déclinaison.

```
private double latitude;
private DeclinationAngle declinationAngle;
private HourAngle hourAngle;
```

La méthode get\_value() de cette class nous calcule la valeur de l'angle de l'élévation.

```
public double get_value() {
    double delta = declinationAngle.get_value();
    double hour1 = hourAngle.get_value();

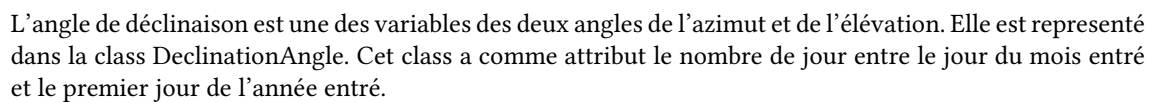
    return (Math.asin(Math.sin(latitude * Math.PI / 180) * Math.sin(delta * Math.PI / 180)
        + Math.cos(latitude * Math.PI / 180) * Math.cos(delta * Math.PI / 180)
        * Math.cos(hour1 * Math.PI / 180)))
        * 180 / Math.PI;
}
```

Cette méthode est basé sur l'équation l'angle de l'élévation qui requiert trois variables : L'angle de déclinaison  $\delta$ , la latitude  $L$  et l'angle horaire. l'équation est :

$$\alpha = \sin^{-1} (\sin L \sin \delta + \cos L \cos \delta \cos H)$$

Comme dans la class AzimuthAngle, on a la méthode print\_value() qui affiche le message "L'angle de l'elevation (alpha) du soleil est : " et la valeur de l'angle d'élévation.

L'angle de déclinaison est l'angle entre les rayons du soleil et l'équateur comme montré dans la figure ci dessous. Sa valeur est comprise entre 23,44 et -23,44 correspondants à l'angle entre le nord géographique et le nord magnétique.



La méthode `get_value()` calcule la valeur de l'angle de déclinaison avec l'équation

Avec (d) le nombre de jours entre le jour du mois entré et le premier jour de l'année entré. Dans le code cet équation est sous forme de la figure ci dessous

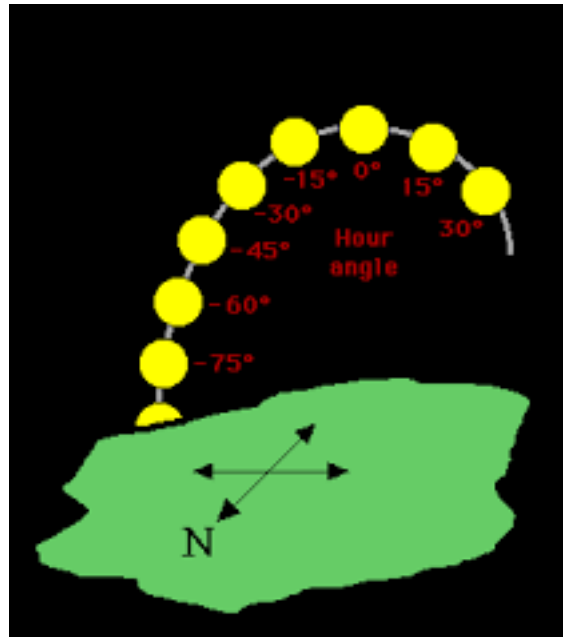
Dans le programme on remplace 10 par 9 car la méthode `daysBetween()` compte le jour entré dans son calcul. On obtient le nombre (d) de jour grâce à des `class` importé.

Ces classes sont utilisée dans la méthode `daysBetween()` afin de calculer (d)

7



## 2.4 L'angle Horaire



L'angle horaire varie de  $15^\circ$  pour chaque heure passer comme montré dans figure au dessus, elle est à  $0^\circ$  à midi solaire. L'angle horaire est la mesure en degrés de la distance entre le soleil à un temps solaire local (lst) et midi solaire. Cet angle converti le temps solaire local en degré avec lequel le soleil bouge dans le ciel. Dans le programme la class HourAngle a comme attribut localSolarTime (lst).

```
public class HourAngle {  
    private LocalSolarTime localSolarTime;
```

L'équation de l'angle horaire a comme variable le temps solaire local (lst), et est sous forme :

$$H = 15(lst - 12)$$

Cette équation est représentée dans le code dans la méthode get\_value

```
public double get_value() {  
    return 15 * (localSolarTime.get_value() - 12);  
}
```

## 2.5 lst

Le temps local solaire elle même une classe qui a comme attribut le temps local (LT) et le time correction factor (TC).

```
private LocalTime localTime;  
private TimeCorrectionFactor timeCorrectionFactor;
```

Les variables de l'équation du temps solaire local sont : le temps local (LT) et le time correction factor (TC). l'équation est  $lst = LT + \frac{TC}{60}$ , elle est représentée dans la méthode `get_value()`

```
public double get_value(){
    return localTime.get_value() + timeCorrectionFactor.get_value()/60;
}
```

### 2.5.1 Temps locale

Le temps local est représenté dans la class `LocalTime`, et a comme attribut l'heure et les minutes qui doivent être entré dans le terminal à l'exécution.

La méthode `get_value` de cet class calcule la somme de l'heure et les minutes, en heures en divisant les minutes par 60.

```
public class LocalTime {
    private int hour;
    private int minute;

    public LocalTime(int hour, int minute) {
        this.hour = hour;
        this.minute = minute;
    }

    public double get_value() {
        return ((double)hour)+(((double) minute)/60);
    }
}
```

### 2.5.2 Time Correction Factor

La classe `TimeCorrectionFactor` contient la longitude (Lg), l'équation du temps (Eot) et UTC qui est le temps de la zone locale, et dont la valeur est entré dans le terminale lors de l'execution du programme.

La méthode `get_value` execute l'équation dont les variables sont : la longitude (Lg), l'équation du temps (Eot) et UTC, et est sous forme de :

$$TC = 4(Lg - 15UTC) + EoT$$

et est représenté dans la figure ci dessous

```

public class TimeCorrectionFactor {
    private double longitude;
    private EquationOfTime equationOfTime;
    private int universalTimeCoordinate;

    public TimeCorrectionFactor(EquationOfTime equationOfTime, double longitude,
        int universalTimeCoordinate) {
        this.longitude = longitude;
        this.universalTimeCoordinate = universalTimeCoordinate;
        this.equationOfTime = equationOfTime;
    }

    public double get_value() {
        return 4 * (longitude - 15 * universalTimeCoordinate) + equationOfTime.get_value();
    }
}

```

### 2.5.3 L'équation du temps

cette équation corrige l'excentricité de l'orbite terrestre et son inclinaison de l'axe. Son équation varie selon B (la fraction année en radians)

$$EoT = 229,18(0.000075 + 0.001868 \cos B - 0.032077 \sin B - 0.014615 \cos 2B - 0.040849 \sin 2B)$$

```

public class EquationOfTime {
    private B b;

    public EquationOfTime(B b) {
        this.b = b;
    }

    public double get_value() {
        return 229.18 * (0.000075 + 0.001868 * Math.cos(b.get_value() )
            - 0.032077 * Math.sin(b.get_value() )
            - 0.014615 * Math.cos(2 * b.get_value() )
            - 0.040849 * Math.sin(2 * b.get_value() ));
    }
}

```

### 2.5.4 B (la fraction année en radians)

Dans la class B on import le calendrier Gregorian pour calculer numberOfDays (nombre de jour), qui est un attribut comme le et localTime (temps local).

```

import java.util.GregorianCalendar;
import java.util.Date;

public class B {
    public int numberOfDays;
    private LocalTime localTime;
}

```

La méthode `daysBetween()` a le même rôle que celle de la classe `DeclinationAngle`, elle calcule le nombre de jours entre le premier jour de l'année entrée et le jour entré au terminale.

```
public int daysBetween(Date d1, Date d2) {
    return (int) ((d2.getTime() - d1.getTime()) / (1000 * 60 * 60 * 24));
}
```

La méthode `get_value` permet de calculer la valeur de  $B$  en fonction de `numberOfDays` ( $d$ ) et de `localtime` ( $LT$ ), en utilisant la fonction suivante :

$$B = \frac{2\pi}{365} \left( d - 1 + \frac{LT-12}{24} \right)$$

```
public double get_value() {
    return (2*Math.PI/365)*(numberOfDays+((localTime.get_value()-12)/24));
}
```

## 2.6 class SolarParameters

La class `SolarParameters` fait passer l'ensemble des objets en paramètres pour une meilleure représentation.

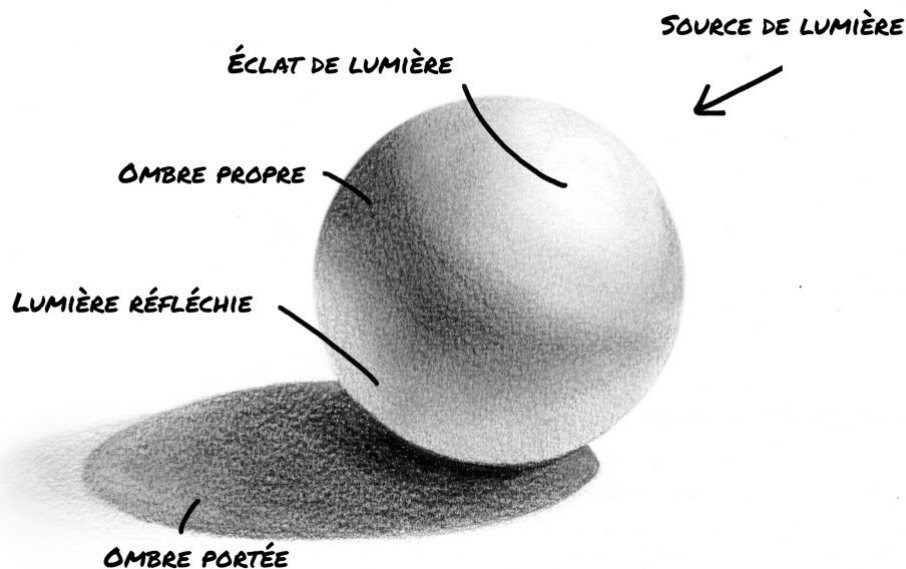
```
public class SolarParameters {

    public B b;
    public EquationOfTime equationOfTime;
    public AzimuthAngle azimuthAngle;
    public ElevationAngle elevationAngle;
    public DeclinationAngle declinationAngle;
    public HourAngle hourAngle;
    public LocalSolarTime localSolarTime;
    public LocalTime localTime;
    public TimeCorrectionFactor timeCorrectionFactor;
    public AngleOfIncidence angleOfIncidence;
    public ShadowReach shadowReach;

    public SolarParameters(int hour, int minute, int jour, int mois, int annee, double latitude, double longitude,
        int universalTimeCoordinate) {
        this.b = new B(hour, minute, jour, mois, annee);
        this.localTime = new LocalTime(hour, minute);
        this.declinationAngle = new DeclinationAngle(jour, mois, annee);
        this.equationOfTime = new EquationOfTime(b);
        this.timeCorrectionFactor = new TimeCorrectionFactor(equationOfTime, longitude, universalTimeCoordinate);
        this.localSolarTime = new LocalSolarTime(localTime, timeCorrectionFactor);
        this.hourAngle = new HourAngle(localSolarTime);
        this.elevationAngle = new ElevationAngle(hourAngle, declinationAngle, latitude);
        this.azimuthAngle = new AzimuthAngle(hourAngle, elevationAngle, declinationAngle, latitude);
        this.angleOfIncidence = new AngleOfIncidence(elevationAngle, azimuthAngle);
        this.shadowReach = new ShadowReach(elevationAngle);
    }
}
```

### 3 La portée d'une ombre

Quand on éclaire un objet ou une personne, une partie de l'objet n'est pas éclairée c'est ce qu'on appelle l'ombre propre, une autre ombre se projette sur une surface, c'est cette ombre que nous appelons l'ombre portée et celle que nous allons étudier. Cette ombre varie en fonction de la position par rapport à la source de lumière.



Pour mesurer la longueur  $L$  de l'ombre portée, on a besoin de la hauteur  $C$  de l'objet illuminée, et de l'angle de l'élévation.

Nous avons :  $\frac{C}{L} = \tan \alpha$

On en conclut que  $L = \frac{C}{\tan \alpha}$

Enfin  $L = \frac{C}{\tan \alpha}$

Dans notre programme, nous avons la class ShadowReach qui a comme attribut elevationAngle

```
private ElevationAngle elevationAngle;
```

La méthode print\_value() de cet class permet d'afficher l'équation de la longueur  $L$  de l'ombre

```
public String print_value() {
    return "la longueur de l'ombre (L)=la hauteur de l'objet (C) x cotang(l'angle de l'elevation (alpha)) \n"+
           + "L=Cx" + 1/Math.tan(elevationAngle.get_value()*Math.PI/180)+"\n*****";
}
```

et donne comme resultat :

```
la longueur de l'ombre (L)=la hauteur de l'objet (C) x cotang(l'angle de l'elevation (alpha))
*****
L=Cx0.5844527713006744
*****
```

## 4 Énergie reçue par une surface

La terre reçoit des different types de rayonnement le rayonnement diffus, direct et globale. Dans cette partie on abordera le rayonnement direct. Le rayonnement direct est le rayonnement venant directement du soleil, est pour calculer son énergie reçue par une surface de pente (p) et d'orientation 0, on a l'équation en fonction de (p) et (i) l'angle d'incidence.

$$S'(p, 0) = i' \cos i$$

Avec  $\cos i = \sin p \cos \alpha \cos (A - A_n) + \cos p \sin \alpha$

Avec  $\alpha$  l'angle de l'élévation, A l'angle de l'azimut, et  $A_n$  l'angle de l'azimut à la normale.

Dans le code on trouve la class AngleOfIncidence, qui a comme attribut azimuthAngle et elevationAngle

```
private ElevationAngle elevationAngle;  
private AzimuthAngle azimuthAngle;
```

Ensuite on a les méthodes permettant d'afficher les équations concernant l'énergie reçue

```
public String get_value() {  
    return "acos(sin(p) x cos(" + azimuthAngle.get_value() + " - An) +cos(p) x "  
        + Math.sin(elevationAngle.get_value() * Math.PI / 180) + " )";  
}  
  
public String getCosvalue() {  
    return "sin(p) x cos(" + azimuthAngle.get_value() + " - An) +cos(p) x "  
        + Math.sin(elevationAngle.get_value() * Math.PI / 180);  
}  
  
public String print_value() {  
    return "S'(la pente (p),Angle azimut à la normal (An)) = derivé de l'angle incident (i')xcos(angle incident (i) ) \n"+  
        + print_cosval() + "\n"+"***** \n"  
        + print_val() + "\n"+"***** \n"  
        + "i' = 1/Math.sqrt(1-i^2) \n"+"***** \n"  
        + "i' = 1/Math.sqrt(1- ( " + get_value() + ")^2) \n"+"***** \n"  
        + "S'(p,An) =1/Math.sqrt(1- ( " + get_value() + ")^2) x ( " + getCosvalue() + " )"+" \n*****  
}  
  
public String print_val() {  
    return "i = acos(sin(p) x cos(alpha) x cos(A-An) + cos(p) x sin(alpha)) \n"+"*****  
        + "i = " + get_value();  
}  
  
public String print_cosval() {  
    return "cos(i)=sin(p) x cos(alpha) x cos(A-An) + cos(p) x sin(alpha) \n"+"*****  
        + "cos(i) = " + getCosvalue();  
}
```

### 4.1 Conclusion

Après execution du programme, le resultat affiché dans le terminale est constitué de l'angle de l'azimut et l'angle d'élévation, qui constituent la position du soleil.

Ainsi que l'équation de la longueur de l'ombre portée en fonction de la hauteur de l'objet éclairé, et de l'angle d'élévation.

Enfin on a l'équation de l'énergie reçu par une surface, en fonction de l'angle de l'incidence dont l'équation est aussi affiché.

Et pour une lisibilité du code j'ai créer la class SolarParameters

## 5 Resume en Anglais

In this project I tried to make a program that when executed, you are asked to enter certain parameters (hour, minutes, day, month, year, latitude, longitude, UTC) and give you the sun's position accordingly. to Execute this program you need to change the directory to **src** in the terminal, and then enter the command **javac Exec.java && java Exec** in order to enter your parameters.

After entering all your parameters, you will have the position of the sun that is based on the azimuth angle which is the angle that is defined in the horizontal plane, and is 0° in the north , at 90° in the east, at 180° in the south, and at 270° or -90° in the west. the sun's position is also dependent on the elevation angle that you can see after the execution of the program, the elevation angle is the angle between the horizontal plane and the sun in the sky.

Then we have the shadow cast, as it name says it is the shadow of an illuminated object that is cast in surface, the program shows the equation that varies depending on the height of the object, and the tangent of the elevation angle.

And finally we have the energy received by surface, the same way as the the shadow cast, the program shows the equation used to calculate the energy with the angle of incidence.

You can see in the picture the result expected

```
L'angle Azimuth (A) est : 140.3591752672196
*****
L'angle de l'elevation (alpha) du soleil est : 59.695731093881164
*****
la longueur de l'ombre (L)=la hauteur de l'objet (C) x cotang(l'angle de l'elevation (alpha))
*****
L=Cx0.5844527713006744
*****
S'(la pente (p),Angle azimut à la normal (An)) = dérivé de l'angle incident (i')xcos(angle incident (i) )
*****
cos(i)=sin(p) x cos(alpha) x cos(A-An) + cos(p) x sin(alpha)
*****
cos(i) = sin(p) x cos(140.3591752672196 - An) +cos(p) x 0.8633579576394883
*****
i = acos(sin(p) x cos(alpha) x cos(A-An) + cos(p) x sin(alpha))
*****
i = acos(sin(p) x cos(140.3591752672196 - An) +cos(p) x 0.8633579576394883 )
*****
i' = 1/Math.sqrt(1-i^2)
*****
i' = 1/Math.sqrt(1- (acos(sin(p) x cos(140.3591752672196 - An) +cos(p) x 0.8633579576394883 ))^2)
*****
S'(p,An) =1/Math.sqrt(1- (acos(sin(p) x cos(140.3591752672196 - An) +cos(p) x 0.8633579576394883 ))^2) x (sin(p) x cos(140.3591752672196 - An) +cos(p)
x 0.8633579576394883)
*****
eiklnot@eiklnot-HP:~/Bureau/project/AzimuthAngle/src$
```