CAPÍTULO 4 – Insolação Terrestre

4.1 Objetivo.

A proposta deste projeto consiste em implementar um aplicativo que calcule o tempo de iluminação terrestre de acordo com o dia, mês e o ano assim como também de acordo com a latitude geográfica no modo GMT (Grau, Minuto e Segundo) e em relação ao o UTC (do inglês, Tempo Coordenado Universal) que se pretende saber a respeito da incidência solar, com isto o sistema deve calcular e mostar os resultados do tempo de insolação na localidade desejada em horas, minutos e segundos, bem como o momento da aurora em hora minuto e segundo e da mesma forma o momento do poente, em hora minuto e segundo. Um outro objetivo, segudário, consiste em testar a máquina do celular em relação a sua precisão em função dos cálculos trigonométricos, como forma de experiência do seu comportamento nessas condições de cálculo.

4.2 Descrição.

O movimento do planeta Terra no espaço marca os períodos de cada um dos seus movimentos, como o período de rotação em seu próprio eixo que é denominado dia, translação ao redor do sol denominado ano e o de precessão e nutação devido a sua formação geodésica com período na ordem, de 41 mil anos, previsto pelo sérvio Milankovitch (1879-1958) em 1920. No entanto, apesar da grande precisão desses movimentos registrados no calendário solar, ao longo dos tempos, em que ocorre o desenvolvimento humano, até agora foram criados diversos sitemas para se medir o tempo e calcular os dias, dentre os diversos sitemas oficialmente no Brasil, assim como em muitos outros países adotam o Calendário Gregoriano. Esse calendário oficialmente instituído pelo Papa Gregório XIII (1502-1585), na Itálha em 24 de fevereiro de 1582, esse calendário também permite expandi-lo para datas anteriores a sua proclamação, assim como para datas antes de Cristo (a.C.) mediante a redução de 1 ano para que se possa calcular o ano bissexto. A discrepância existente entre o calendário solar e outros calendários consiste em que o calendário solar trabalha com números fracionários e os calendários comumente utilizados utilizam números inteiros, que são comumente relacionados ao trabalho, devido a isto ainda não temos um calendário civil perfeito, mas que necessita de correções de tempos em tempos, mas que atende satisfatóriamente as necessidades do cotidiano da maioria das atividades humanas seculares e este projeto consiste em converter uma data do calendário civil gregoriano para uma relação com os movimentos de rotação da Terra e translação dela em torno do Sol em determinada data específica e com esse processo retornar a hora, minuto e segundo de duração da insolação da luz do Sol, assim como da hora, minuto e segundos da aorora, “o necer do sol”, bem como do poente, quando osol não fica mais visível no local indicado pelo fuso da hora local, UTC (Tempo Coordenado Universal), e pela latitude geográfica em grau, minuto e segundo.

O funcionamento desse sistema tem três etapas que são: entrada de dados, cálculos e exibição dos resultados.

Na primeirs etapa o usuário dará entrada dos dados solicitados pelo sistema e o sistema verifica sua consistência a cada entrada, solicitando que repita a entrada caso os dados não sejam válidos, sendo que o ano fica limitado a 2418 a.C. e 5582, devido a correções que precisarão ser aperfeiçoadas no calendário gregoriano para datas maiores que 4 milhênios, a entrada do mês fica condicionado, naturalmente a entradas entre 1 e 12 correspondente aos mês do ano, da mesmo forma o dia, mas este poderão variar conforme o mês informado e se o ano for bissexto calculado pela fórmula 1 do item 4.3, o grau da latitude, será aceito se compreender valores entre -90° e +90°, os minutos entre -60’ e +60’ e os segundos entre -60’’ e +60’’ e o valor do fuso horário local UTC em relação a hora GMT nãp poderão ser diferentes dos valores compreendidos entre -12 a +12 em relação ao fuso GMT (fuso 0, de referência).

Os cálculos na segunda etapa são realizados conforme fórmulas exibidas no item 4.3, como se segue: Calcula-se o total de dias corridos do inicio do ano civil até o dia do mês através das formula 1 e 7, depois converte-se a inclinação do equador terrestre em relação ao equador solar do modo GMT para o mode decimal pela fórmula 2, em seguida faze-se o cálculo da duração da insolação pela fórmula 3, então pode-se calcular a hora da aurora com a fórmula 4 e a hora do poente com a fórmula 5.

Na terceira e última etapa do provesso desse sistema, a formula 6 permitirá a conversão do modo decimal dos resultados obtidos na segunda etapa para o modo GMS, que é melhor compreendido pelo usuário, passando de valores decimal para o formato de grau, minuto e segundo, finalizando assim o processo para se conhecer a duração da insolação na Terra, os momentos da aurora e do poente do dia que se pretende saber.

4.3 Fórmulas (matemáticas) e Tabelas.

- Fórmula 1 - Resolve se o ano é bissexto:

$$p: a > 0 \quad [1]$$
$$ac = ano - (1 * \neg p) \quad [2]$$
$$q: ac \equiv 0\ mod(4) \quad [3]$$
$$r: ac \not\equiv 0\ mod(100) \quad [4]$$
$$s: ac \equiv 0\ mod(400) \quad [5]$$
$$ab: q \wedge (r \vee s) \quad [6]$$

$onde$:
$p, q, r, s = valor\ lógico$ {0.1}
$a = ano\ entre\ -2418\ e\ 5582$
$ac = ano\ corrigido$
$ab = ano\ bissexto$ valor lógico {0,1}

- Fórmula 2 - Converte GSM (Grau, Minuto e Segundo) para grau decimal:

$$d = \mathrm{g} + \mathrm{m}/60 + \mathrm{s}/60/60 \quad [7]$$

Onde:
g = Grau
m = Minuto
s = Segundo
d = grau decimal

- Formula 3 - Resolve de forma decimal o tempo de insolação terrestre:

$$rad = \frac{\pi}{180} \quad [8]$$

$$grd = \frac{180}{\pi} \quad]9]$$

$$a = -\tan(rad * l) \quad [10]$$

$$b = rad * ie$$

$$c = \frac{360}{dAC} * (eq * dst) \quad [11]$$

$$d = \sin(rad * c) \quad [12]$$

$$e = \tan(b * d) \quad [13]$$

[14]

Onde:
rad = coeficiente de graus para radiano;
grd = coeficiente de radiano oara graus;
l = coordenada de latitude em decimal;
ie = Inclinação do equador terrestre em relação ao equador do Sistema Solar;
dAC = Número de dias do Ano Civil;
eq = Número de dias em relação ao equinócio;
dst = Número de dias corridos referente ao dia de insolação terrestre calculado;
a, b, c, d, e = variáveis auxiliares.

- Forma 4 - Resolve de forma decimal a hora da aurora para um determinado dia do ano:

$$\text{ha} = 12 - \frac{it}{2} \quad [15]$$

Onde:
it = Insolação Terrestre fórmula [14]
ha = hora da aurora

- Forma 5 - Resolve de forma decimal a hora do poente para um determinado dia do ano:

$$\text{hp} = 12 + av = 12 - \frac{it}{2} \quad [16]$$

Onde:
it = Insolação Terrestre fórmula [14]
hp = hora do poente

- Formula 6 - Converte modo de tempo em decimal para modo GMS:

$$D(x) = x - \lfloor x \rfloor \quad [16]$$
$$Int(x) = x - D(x) \quad [17]$$
$$h = Int(hd) \quad [18]$$
$$md = D(hd) * 60 \quad [19]$$
$$m = Int(md) \quad [20]$$
$$s = D(md) * 60 \quad [21]$$

Onde:
$x = Qualquer\ valor\ real \geq 0$
$h = Hora$
$md = minuto\ decimal$
$m = Minuto$
$s = Segundo$
$hd = Hora\ em\ decimal \geq 0$

- Formula 7 – Calcula o total de dias corridos do inicio do ano civil até o dia do mês

$$(\text{m} = 1) \leftrightarrow (\text{ds} = 0) \quad [22]$$
$$(\text{m} = 2) \leftrightarrow (\text{ds} = 31)$$

(m = 3) ↔ (ds = 59)
(m = 4) ↔ (ds = 90)
(m = 5) ↔ (ds = 120)
(m = 6) ↔ (ds = 151)
(m = 7) ↔ (ds = 181)
(m = 8) ↔ (ds = 212)
(m = 9) ↔ (ds = 243)
(m = 10) ↔ (ds = 273)
(m = 11) ↔ (ds = 304)
(m = 12) ↔ (ds = 334)
dsp = ds + d [23]
((ab = 1) ∧ (m > 2)) ↔ (dst = dsp + 1) [24]
(ab = 0) ↔ (dst = dsp) [25]

Onde:
$m = mês$
$d = dia\ do\ mês$
$ds = dias\ acumulado$
$dsp = dias\ parcial\ acumulado$
$dst = dias\ total\ acumulado$

4.4 Problema Contextualizado.

Os cálculos realizados abaixo, são referentes a memória de cálculo do 1º Teste no algoritmo exibido no item 4.5 e das telas em 4.6, os quais os valores de entradas foram tomando como premissas para a elaboração a seguir, sendo: dia 15 de setembro de 2016, na latitude -22º54'10'' e no fuso horário -3 UTC:

- Cálculo de dias corrido do ano civil em relação ao dia do mês, ver fórmulas 1 e 7 do item 4.3.

  A - Determinação se o ano é bissexto conforme a formula 1 para 2016:

  1º) $p: 2016 > 0\ \therefore p(V)\ tal\ que\ p = 1\ \ eq.[1]$
  2º) $ac = 2016 - (1 * \neg 1) \therefore ac = 2016\ eq.[2]$
  3º) $q: 2016 \equiv 0\ mod(4)\ \therefore q(V)\ tal\ que\ q = 1\ eq.[3]$
  4º) $r: 2016\ \not\equiv 0\ mod(100) \therefore r(V)\ tal\ que\ r\ =\ 1\ eq.[4]$
  5º) $s: 2016\ \equiv 0\ mod\ (400)\ \therefore\ s(F)\ tal\ que\ s\ =\ 0\ \ eq.[5]$
  6º) $ab: q \wedge (r \vee s) \therefore\ ab(V)\ tal\ que\ ab\ =\ 1\ \ eq.[6]$
  Logo, 2016 é um ano bissexto.

  B – Determinação do número de dias corridos do início do ano civil até o dia 15 do mês de setembro, pela formula 7:

1º) (m = 9) ↔ (ds = 243) eq. [22]
2º) dsp = 243 + 15 ∴ dsp = 258 eq. [23]
3º) ((ab = 1) ⋀ (m>2)) ↔ (dst = 258 + 1) ∴ dst = 259 eq. [24]
Logo, são 259 dias corridos do dia 1º de janeiro até 15 de setembro de 2016.

- Cálculo de conversão da inclinação do equador terrestre em relação ao equador solar do modo GMT para o mode decimal da posição oficial da cidade do Rio de Janeiro com latitude 22° 54' 10'' S, formula 2.

$$d = -22 + \frac{54}{60} + \frac{\frac{10}{60}}{60} \therefore d \cong -21.097 \ eq.[7]$$

- Cálculo de duração da insolação pela fórmula 3 com as premissas fornecidas.
Derterminação da inclinação do eixo da Terra em relação ao equador do sitema solar:

$$ie = 23 + \frac{27}{60} \therefore ie = 23.45 \ eq.[7]$$

Determinação da duração da insolação na Terra no dia 15 de setembro de 2016:

$$a = -\tan\left(\frac{\pi}{180} * -21.097\right) \therefore a \cong 0.386 \ eq.[10]$$

$$e = \tan\left(\frac{\pi}{180} * 23.45 * \sin\left(\frac{\pi}{180} * \frac{360}{366} * (284.5 * 259)\right)\right) \therefore$$

$$e \cong 0.380 eq.[13]$$

$$it = \frac{2}{15}\cos^{-1}(0.386 * 0.380)\frac{180}{\pi} \therefore it \cong 11.887 \ eq.[14]$$

- Cálculo da hora da aurora, fórmula 4:

$$ha = 12 - \frac{11.887}{2} \therefore ha \cong 6.056 \ eq.[15]$$

- Cálculo da hora do poente com a fórmula 5:

$$hp = 12 + \frac{11.887}{2} \therefore ha \cong 17.943\ eq.[16]$$

- Cálculo de conversão do modo decimal de duração da insolação para o modo GMS, grau, minuto e segundo pela formula 6:

$D(x) = 11.887 - \lfloor 11.887 \rfloor \therefore D(x) = 0.887$ eq. [16]
$Int(x) = 11.887 - 0.887 \therefore Int(x) = 11$ eq. [17]
$h = Int(11.887) \therefore h = 11$ eq. [18]
$md = 0.887 * 60 \therefore$ md $\cong 53.172$ eq. [18]
$m = Int(53.172) \therefore m = 53$ eq. [19]
$s = D(53.172) * 60 \therefore s = 10.35$ eq. [20]

- Cálculo de conversão do modo decimal da aurora para o modo GMS, grau, minuto e segundo pela formula 6:

$D(x) = 6.056 - \lfloor 6.056 \rfloor \therefore D(x) = 0.056$ eq. [16]
$Int(x) = 6.056 - 0.056 \therefore Int(x) = 6$ eq. [17]
$h = Int(6.056) \therefore h = 6$ eq. [18]
$md = 0.056 * 60 \therefore$ md $\cong 3.413$ eq. [18]
$m = Int(3.413) \therefore m = 3$ eq. [19]
$s = D(3.413) * 60 \therefore s = 24.82$ eq. [20]

- Cálculo de conversão do modo decimal do poente para o modo GMS, grau, minuto e segundo pela formula 6:

$D(x) = 17.438 - \lfloor 17.438 \rfloor \therefore D(x) = 0.438$ eq. [16]
$Int(x) = 17.438 - 0.438 \therefore Int(x) = 17$ eq. [17]
$h = Int(17.438) \therefore h = 17$ eq. [18]
$md = 0.438 * 60 \therefore$ md $\cong 56.586$ eq. [18]
$m = Int(56.586) \therefore m = 56$ eq. [19]
$s = D(56.586) * 60 \therefore s = 35.17$ eq. [20]

4.5 Algoritmo.

4.5.1 Visualg

```
algoritmo "InsolacaoTerrestre"
// Função : Calcula Insolação Terrestre
// Autor : Sérgio da Silva Pereira
// Data : 18/09/2016

// Seção de Declarações
```

```
var
  exit: logico

   // Função Ajustar UTC
   funcao utcToStr (nUTC: real): caracter
   var
      str: caracter
   inicio

      se (nUTC > 0) OU (nUTC = 0) entao
         str <- numpcarac(nUTC)
         se (Compr(str) = 1) entao
            str <- "+0" + str
         senao
            str <- "+" + str
         fimse
      senao
         str <-  numpcarac(Abs(nUTC))
         se (Compr(str) = 1) entao
            str <- "-0" + str
         senao
            str <- "-" + str
         fimse
      fimse

      retorne(str)
   fimfuncao

   // Função verifica se o ano é bissesto
   funcao bissesto (a: inteiro): logico
   var
      bi: logico
      ac: inteiro

   inicio
      se a < 1 entao
        ac <- a-1
      senao
        ac <- a
      fimse

      se (((ac MOD 4)=0) e ((ac MOD 100)<>0)) OU ((ac MOD 400)=0) entao
         bi <- VERDADEIRO
      senao
        bi <- FALSO
      fimse

      retorne(bi)
   fimfuncao


   //Função converte o número referente ao mês em texto
   funcao mesToStr (m: inteiro): caracter
   var
      str: caracter

   inicio
      escolha (m)
         caso 1
              str <- "janeiro"
         caso 2
              str <- "fevereiro"
         caso 3
              str <- "março"
         caso 4
              str <- "abril"
         caso 5
              str <- "maio"
         caso 6
              str <- "junho"
         caso 7
              str <- "julho"
         caso 8
              str <- "agosto"
         caso 9
              str <- "setembro"
         caso 10
```

```
            str <- "outubro"
      caso 11
            str <- "novembro"
      caso 12
            str <- "dezembro"
   fimescolha

   retorne(str)
fimfuncao

// Função que calcula o numero de dias corridos no ano
funcao diaDoAno(d, m: inteiro; bi: logico): inteiro
var
   dias: inteiro
   diasAno: vetor [1..12] de inteiro

inicio
   diasAno[1]  <- 0
   diasAno[2]  <- 31
   diasAno[3]  <- 59
   diasAno[4]  <- 90
   diasAno[5]  <- 120
   diasAno[6]  <- 151
   diasAno[7]  <- 181
   diasAno[8]  <- 212
   diasAno[9]  <- 243
   diasAno[10] <- 273
   diasAno[10] <- 304
   diasAno[12] <- 334

   se ((bi = VERDADEIRO) E (m > 2))  entao
      dias <- diasAno[m] + d + 1
   senao
      dias <- diasAno[m] + d
   fimse

   retorne(dias)
fimfuncao

// Função que informa quantos dias tem um determinado mês
funcao diasMes(ms: inteiro; bi: logico): inteiro
var
   dias: inteiro
   diasDoMes: vetor [1..12] de inteiro

inicio

   diasDoMes[1]  <- 31
   diasDoMes[2]  <- 28
   diasDoMes[3]  <- 31
   diasDoMes[4]  <- 30
   diasDoMes[5]  <- 31
   diasDoMes[6]  <- 30
   diasDoMes[7]  <- 31
   diasDoMes[8]  <- 31
   diasDoMes[9]  <- 30
   diasDoMes[10] <- 31
   diasDoMes[11] <- 30
   diasDoMes[12] <- 31

   se ((bi = VERDADEIRO) E (ms = 2)) entao
      dias <- diasDoMes[ms] + 1
   senao
        dias <- diasDoMes[ms]
   fimse

   retorne(dias)
fimfuncao

// Função que converte grau, minuto e segundo em decimal
funcao gmsToDec(g, m, s: real): real
var
  decimal: real

inicio
   retorne(g+m/60+s/60/60)
fimfuncao
```

```
   // Funçao que converte a hora parrasda em decimal para hora, minuto e segundo
   funcao timeToStr (time: real): caracter
   var
      strHora, strMinuto, strSegundo: caracter

   inicio
      strHora <- numpcarac(int(time))+"h "
      strMinuto <- numpcarac(int((time-int(time))*60))+"min "
      strSegundo <- Copia(numpcarac(((time-int(time))*60-int((time-
int(time))*60))*60),0,5)+"s")

      retorne(strHora + strMinuto + strSegundo)
   fimfuncao

   // Procedimento que plota o resultado para o usuário
   procedimento saida(d: inteiro; m: caracter; a: inteiro; hi, ha, hp: real;
sUTC: caracter)
   inicio
      escreval(" ")
      escreval(" ")
      escreval("RESULTADO - INSOLAÇÃO TERRESTRE EM "+numpcarac(d)+" DE
"+Maiusc(m)+" DE "+numpcarac(a)+":")
      escreval("============================================================")
      escreval("Duração prevista de "+timeToStr(hi)+" "+sUTC+" UTC;")
      escreval("Aurora prevista às "+timeToStr(ha)+" "+sUTC+" UTC;")
      escreval("Poente previsto às "+timeToStr(hp)+" "+sUTC+" UTC;")
      escreva ("============================================================")
      escreval(" ")
   fimprocedimento

   // Função de controle lógico do aplicativo
   funcao execInsolacoTerrestre (): logico
   var
      // Variáveis de Valores de Entrada
      dia, mes, ano: inteiro
      latitudeGraus, latitudeMinutos, latitudeSegundos, UTC: real

      // Variáveis de valores internos
      grausEquadorSolar, minutosEquadorSolar, segundosEquadorSolar: real
      equinocio: real
      mesStr, utcStr: caracter

      // Variáveis de Valores Calculados Auxiliares
      fBissesto: logico
      diasAnoCivil, diasX: real
      equadorSolar, latitude: real

      // Variáveis de Valores Calculados de Saída
      horasDeInsolacao, HoraAurora, HoraPoente: real

      // variável de opção do usuário
      encerrar: caracter
      sair: logico
      str: caracter

   inicio
      // Seção de Comandos


      // Parâmetros Iniciais
      grausEquadorSolar <- 23;
      minutosEquadorSolar <- 27;
      segundosEquadorSolar <- 0;
      equinocio <- 284.5;

      dia <- 0
      mes <- 0
      ano <- 0

      latitudeGraus <- 0
      latitudeMinutos <- 0
      latitudeSegundos <- 0
      UTC <- 0

      mesStr <- ""
      utcStr <- ""
```

```
fBissesto <- FALSO
diasAnoCivil <- 0
diasX <- 0

equadorSolar <- 0
latitude <- 0

horasDeInsolacao <- 0
HoraAurora <- 0
HoraPoente <- 0

encerrar <- ""
sair <- FALSO

// Dados de Entrada do Usuário
limpatela
escreval("============================================================")
escreval("                  INSOLAÇÂO TERRESTRE")
escreval("============================================================")

// Seguem filtros para tratamento das entradas

repita
   escreva("Digite o Ano [-2418 a 5582]: ")
   leia(ano) // 2016
ate (ano > -2419) E (ano < 5583)

escreval(" ")

fBissesto <- bissesto(ano)

repita
   escreva("Digite o Mês [1 a 12]: ")
   leia(mes) // 9
ate ((mes > 0) E (mes < 13))

escreval(" ")

str <- "Digite o Dia [1 a "+numpcarac(diasMes(mes, fBissesto))+"]: "

repita
   escreva(str)
   leia(dia) // 20
ate ((dia >= 1) E (dia <= diasMes(mes, fBissesto)))

escreval(" ")

repita
   escreva("Digite o Grau da Latitude: ")
   leia(latitudeGraus)    // -23
ate (latitudeGraus > -90) E (latitudeGraus < 90)

escreval(" ")

repita
   escreva("Digite o Minuto da Latitude: ")
   leia(latitudeMinutos)  // 0
ate (latitudeMinutos >= 0) E (latitudeMinutos < 60)


escreval(" ")

repita
   repita
      escreva("Digite o Segundo da Latitude: ")
      leia(latitudeSegundos) // 0
   ate (latitudeSegundos >= 0) E (latitudeSegundos < 61)
ate ((latitudeGraus <> 0) OU (latitudeMinutos <> 0) OU (latitudeSegundos <>
0))

escreval(" ")

repita
   escreva("Digite o UTC do Fuso: ")
   leia(UTC) // -3
ate ((UTC >= -12) E (UTC <= 12))
```

```
      // Indicação de hora local em relação a hora GMT, DST não está previsto
      utcStr <- utcToStr(UTC)

      // Ano Civil
      se fBissesto entao
         diasAnoCivil <- 366
      senao
           diasAnoCivil <- 365
      fimse

      // Ajustar o Mês por Extenso
      mesStr <- mesToStr(mes)

      // Conta dias no ano
      diasX <- diaDoAno(dia, mes, fBissesto)

      // Concatenar e Ajustar Valores
      equadorSolar <- gmsToDec(grausEquadorSolar, minutosEquadorSolar,
segundosEquadorSolar)
      latitude <- gmsToDec(latitudeGraus, latitudeMinutos, latitudeSegundos)

      // Ajuste da latitude para evitar error de divisão por zero na função
      se (latitude = 0) entao
         latitude <- 0.000000001) // Valor suficiente para ser insignificante na
ordem de grandeza nas dimenções para latitude terrestre
      fimse

      // Cálculos de Insolação, Auróra e Poente
      horasDeInsolacao <- ((2/15)*ArcCos(-
Tan(Pi/180*latitude)*Tan(Pi/180*equadorSolar*Sen(Pi/180*(360/diasAnoCivil*(equino
cio+diasX)))))*180/Pi)
      HoraAurora <- 12-((2/15)*ArcCos(-
Tan(Pi/180*latitude)*Tan(Pi/180*equadorSolar*Sen(Pi/180*(360/diasAnoCivil*(equino
cio+diasX)))))*180/Pi)/2
      HoraPoente <- 12+((2/15)*ArcCos(-
Tan(Pi/180*latitude)*Tan(Pi/180*equadorSolar*Sen(Pi/180*(360/diasAnoCivil*(equino
cio+diasX)))))*180/Pi)/2

      // Saida
      saida(dia, mesStr, ano, horasDeInsolacao, HoraAurora, HoraPoente, utcStr)

      // Tratamento para fim do app ou para repetir a consulta
      escreval("Digite 'S' para Sair,")
      escreva ("ou 'qualquer outra tecla' para reiniciar: ")
      leia(encerrar)
      limpatela

      se (encerrar = "S") ou (encerrar = "s") entao
         sair <- VERDADEIRO
      senao
           sair <- FALSO
      fimse

      retorne(sair)
   fimfuncao

// Escopo principal
inicio

   // loop que permite refazer a consulta
   repita
      exit <- execInsolacoTerrestre ()
   ate (exit = VERDADEIRO)

fimalgoritmo
```

4.5.2 Java Console

```
import java.util.Scanner;

public class InsolacaoTerrestre {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
```

```
            execInsolacoTerrestre ();
    }

    // Método Ajustar UTC
private static String utcToStr (double timeUTC) {

       String utcStr;

    // Ajustar timeUTC
    if (Double.compare(timeUTC, 0.0)  >= 0) {
        utcStr = String.valueOf(timeUTC);

        if (utcStr.length() == 1) {
            utcStr = "+0" + utcStr;
        } else {
            utcStr = "+" + utcStr;
        }
    } else {
        utcStr = String.valueOf(Math.abs(timeUTC));

        if (utcStr.length() == 1) {
            utcStr = "-0" + utcStr;
        } else {
            utcStr = "-" + utcStr;
        }
    }

    return utcStr;
}

// Método verifica se o ano é bissesto
private static boolean bissesto (int a) {

    boolean bissesto; //bi: logico
    int ac; //: inteiro

    if (a < 1){
       ac = a-1;
    }else{
       ac = a;
    }

    if ((((ac % 4) == 0) && ((ac % 100) != 0)) || ((ac % 400) == 0)) {
       bissesto = true;
    } else {
       bissesto = false;
    }

    return bissesto;
}


//Método converte o número referente ao mês em texto
private static String mesToStr (int m) {

    String str = "";

    switch (m){
           case 1:
               str = "janeiro";
                    break;
           case 2:
                   str = "fevereiro";
                   break;
           case 3:
                   str = "março";
                   break;
           case 4:
                   str = "abril";
                   break;
           case 5:
               str = "maio";
                   break;
              case 6:
               str = "junho";
                   break;
```

```
            case 7:
             str = "julho";
                break;
            case 8:
             str = "agosto";
                break;
            case 9:
                str = "setembro";
                break;
        case 10:
                str = "outubro";
                break;
        case 11:
                str = "novembro";
                break;
        case 12:
                str = "dezembro";
                break;
    }

    return str;

}

// Método que calcula o numero de dias corridos no ano
private static int diaDoAno(int d, int m, boolean bi){

    int dias;
    int diasAno[] = new int[12];

    diasAno[0]  = 0;
    diasAno[1]  = 31;
    diasAno[2]  = 59;
    diasAno[3]  = 90;
    diasAno[4]  = 120;
    diasAno[5]  = 151;
    diasAno[6]  = 181;
    diasAno[7]  = 212;
    diasAno[8]  = 243;
    diasAno[9]  = 273;
    diasAno[10] = 304;
    diasAno[11] = 334;

    if ((bi = true) && (m > 2)) {
       dias = diasAno[m-1] + d + 1;
       } else {
         dias = diasAno[m-1] + d;
       }

    return dias;
}

// Método que informa quantos dias tem um determinado mês
private static int diasMes(int ms, boolean bi) {

    int dias;
    int diasDoMes[] = new int[12];

    diasDoMes[0]  = 31;
    diasDoMes[1]  = 28;
    diasDoMes[2]  = 31;
    diasDoMes[3]  = 30;
    diasDoMes[4]  = 31;
    diasDoMes[5]  = 30;
    diasDoMes[6]  = 31;
    diasDoMes[7]  = 31;
    diasDoMes[8]  = 30;
    diasDoMes[9]  = 31;
    diasDoMes[10] = 30;
    diasDoMes[11] = 31;

    if ((bi = true) && (ms > 1)) {
       dias = diasDoMes[ms-1] + 1;
    } else {
       dias = diasDoMes[ms-1];
    }
```

```
        return dias;
    }

    // Método que converte grau, minuto e segundo em decimal
    private static double gmsToDec(double g, double m, double s) {

        //double decimal;

        return g+m/60+(s/60)/60;

    }

    // Método que converte a hora parrasda em decimal para hora, minuto e segundo
    private static String timeToStr (double time) {

        String strHora, strMinuto, strSegundo;

        strHora = String.valueOf((int) time) + "h ";
        strMinuto = String.valueOf((int)((time-((int)time))*60)) + "min ";
        strSegundo = String.format("%.2f",((time-(int)time)*60-((int)((time-
(int)time)*60)))*60)+"s";

        return strHora + strMinuto + strSegundo;

    }

    // Método que plota o título do app na interface
    private static void head(){


        System.out.println("=============================================================
======");
         System.out.println("                    INSOLAÇÃO TERRESTRE");

System.out.println("============================================================")
;
    }

    // Método que plota o resultado para o usuário
    private static void saida(int d, String m, int a, double hi, double ha,
double hp, String sUTC) {

            //System.out.println(" ");
            //System.out.println(" ");
            System.out.println("RESULTADO - INSOLAÇÃO TERRESTRE EM
"+String.valueOf(d)+" DE "+m.toUpperCase()+" DE "+String.valueOf(a)+":");

System.out.println("============================================================")
;
            System.out.println("Duração prevista de "+timeToStr(hi)+";");
            System.out.println("Aurora prevista às "+timeToStr(ha)+" "+sUTC+"
UTC;");
            System.out.println("Poente previsto às "+timeToStr(hp)+" "+sUTC+"
UTC;");

System.out.println("============================================================")
;
            //System.out.println(" ");
    }

    private static int anoInput (Scanner entrada_) {

        int ano_;

            do {
                 System.out.println("Digite o Ano [-2418 a 5582]: ");
               ano_ = Integer.parseInt(entrada_.nextLine()); // 2016
            } while ((ano_ < -2418) || (ano_ > 5582));

            return ano_;

    }

    private static int mesInput (Scanner entrada_) {

            int mes_;
```

```
            do {
                  System.out.println("Digite o Mês [1 a 12]: ");
               mes_ = Integer.parseInt(entrada_.nextLine()); // 9
            } while ((mes_ < 1) || (mes_ > 12));

            return mes_;

    }

    private static int diaInput(Scanner entrada_, int mes_, boolean fBissesto_){

            String str = "Digite o Dia [1 a "+Integer.toString(diasMes(mes_,
fBissesto_))+"]: ";
            int dia_;

            do {
                  System.out.println(str);
               dia_ = Integer.parseInt(entrada_.nextLine()); // 20
            } while ((dia_ < 1) || (dia_ > diasMes(mes_, fBissesto_)));

            return dia_;

    }

    private static double latitudeGrausInput (Scanner entrada_) {

            double latitudeGraus_;
            do {
                  System.out.println("Digite o Grau da Latitude: ");
               latitudeGraus_ = Double.parseDouble(entrada_.nextLine());
            } while ((latitudeGraus_ < -90) || (latitudeGraus_ > 90));

            return latitudeGraus_;

    }

    private static double latitudeMinutosInput (Scanner entrada_) {

            double latitudeMinutos_;

            do {
                  System.out.println("Digite o Minuto da Latitude: ");
               latitudeMinutos_ = Double.parseDouble(entrada_.nextLine());  // 0
            } while ((latitudeMinutos_ < 0) || (latitudeMinutos_ > 59));

            return latitudeMinutos_;
    }

    private static double latitudeSegundosInput(Scanner entrada_, double
latitudeGraus_, double latitudeMinutos_) {

            double latitudeSegundos_;

            do {
               do {
                  System.out.println("Digite o Segundo da Latitude: ");
                  latitudeSegundos_ = Double.parseDouble(entrada_.nextLine()); //
0
               } while ((latitudeSegundos_ < 0) || (latitudeSegundos_ > 59));
            } while ((latitudeGraus_ == 0) && (latitudeMinutos_ == 0) &&
(latitudeSegundos_ == 0));

            return latitudeSegundos_;

    }

       private static int UTCinput (Scanner entrada_) {

               int UTC_;

               do {
                  System.out.println("Digite o UTC do Fuso: ");
                  UTC_ = Integer.parseInt(entrada_.nextLine()); // -3
            } while ((UTC_ < -12) && (UTC_ > 12));

            return UTC_;
    }
```

```
    // Método de controle lógico do aplicativo
    private static void execInsolacoTerrestre () {
       // Variáveis de Valores de Entrada
       int dia, mes, ano;
       double latitudeGraus, latitudeMinutos, latitudeSegundos, UTC;

       // Variáveis de valores internos
       double grausEquadorSolar, minutosEquadorSolar, segundosEquadorSolar;
       double equinocio;
       String mesStr, utcStr;

       // Variáveis de Valores Calculados Auxiliares
       boolean fBissesto;
       int diasAnoCivil, diasX;
       double equadorSolar, latitude;

       // Variáveis de Valores Calculados de Saída
       double horasDeInsolacao, HoraAurora, HoraPoente;

       // variável de opção do usuário
       String str;

       // Comandos


       // Parâmetros Iniciais
       grausEquadorSolar = 23;
       minutosEquadorSolar = 27;
       segundosEquadorSolar = 0;
       equinocio = 284.5;

       dia = 0;
       mes = 0;
       ano = 0;

       latitudeGraus = 0;
       latitudeMinutos = 0;
       latitudeSegundos = 0;
       UTC = 0;

       mesStr = "";
       utcStr = "";

       fBissesto = false;
       diasAnoCivil = 0;
       diasX = 0;

       equadorSolar = 0;
       latitude = 0;

       horasDeInsolacao = 0;
       HoraAurora = 0;
       HoraPoente = 0;

       head();

       Scanner entrada = new Scanner(System.in);

       ano = anoInput(entrada);
       fBissesto = bissesto(ano);
       mes = mesInput(entrada);
       dia = diaInput(entrada, mes, fBissesto);
       latitudeGraus = latitudeGrausInput(entrada);
       latitudeMinutos = latitudeMinutosInput(entrada);
       latitudeSegundos = latitudeSegundosInput(entrada, latitudeGraus,
latitudeMinutos);
       UTC = UTCinput(entrada);

       entrada.close();

       // Indicação de hora local em relação a hora GMT, DST não está previsto
       utcStr = utcToStr(UTC);

       // Ano Civil
       if (fBissesto = true){
          diasAnoCivil = 366;
```

```
            } else {
                  diasAnoCivil = 365;
            }

        // Ajustar o Mês por Extenso
        mesStr = mesToStr(mes);

        // Conta dias no ano
        diasX = diaDoAno(dia, mes, fBissesto);

        // Concatenar e Ajustar Valores
        equadorSolar = gmsToDec(grausEquadorSolar, minutosEquadorSolar,
segundosEquadorSolar);
        latitude = gmsToDec(latitudeGraus, latitudeMinutos, latitudeSegundos);

        // Ajuste da latitude para evitar error de divisão por zero na função
        if (latitude == 0) {
           latitude = 0.000000001; // Valor suficiente para ser insignificante na
ordem de grandeza nas dimenções para latitude terrestre
        }


        // Cálculos de Insolação, Auróra e Poente
        horasDeInsolacao =  ((2.0 / 15.0) * Math.acos(-Math.tan(Math.PI / 180.0 *
latitude) *
                    Math.tan(Math.PI / 180.0 * equadorSolar * Math.sin(Math.PI / 180.0
* (360.0 /
                    diasAnoCivil * (equinocio + diasX))))) * 180.0 / Math.PI);

        HoraAurora = 12.0 - ((2.0 / 15.0) * Math.acos(-Math.tan(Math.PI / 180.0 *
latitude) *
                    Math.tan(Math.PI / 180.0 * equadorSolar * Math.sin(Math.PI / 180.0
* (360.0 /
                    diasAnoCivil * (equinocio + diasX))))) * 180.0 / Math.PI) / 2.0;

        HoraPoente = 12.0 + ((2.0 / 15.0) * Math.acos(-Math.tan(Math.PI / 180.0 *
latitude) *
                    Math.tan(Math.PI / 180.0 * equadorSolar * Math.sin(Math.PI / 180.0
* (360.0 /
                    diasAnoCivil * (equinocio + diasX))))) * 180.0 / Math.PI) / 2.0;

        // Saida
        saida(dia, mesStr, ano, horasDeInsolacao, HoraAurora, HoraPoente, utcStr);

    }
}
```

4.5.3 Tela do Aplicativo (View)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="${relativePackage}.${activityClass}" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="35dp" >

        <TextView
            android:id="@+id/textView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Data da Insolação"
            android:textAppearance="?android:attr/textAppearanceLarge" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/textView5"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="Ano [-2000 a 4500]:"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <EditText
            android:id="@+id/editText1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="numberSigned" >

            <requestFocus android:layout_width="wrap_content" />

        </EditText>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Mês [1 a 12]:"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <EditText
            android:id="@+id/editText2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="0.39"
            android:ems="10"
            android:inputType="number" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="50dp" >

            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Dia [1 a 31*]:"
                android:textAppearance="?android:attr/textAppearanceMedium" />

            <EditText
                android:id="@+id/editText3"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:ems="10"
                android:inputType="number" />
        </LinearLayout>

        <TextView
            android:id="@+id/textView11"
            android:layout_width="wrap_content"
            android:layout_height="50dp"
            android:layout_weight="1"
            android:text="(*) O último dia do mês pode variar em função do mês e
do ano a que se refere."
            android:textAppearance="?android:attr/textAppearanceSmall" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
```

```
    android:layout_height="35dp" >

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Local da Insolação"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Latitude (GMS):"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/textView7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Grau:"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <EditText
           android:id="@+id/editText4_"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="numberSigned" />


    <TextView
        android:id="@+id/textView8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Minuto:"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <EditText
        android:id="@+id/editText5_"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="number" />

    <TextView
        android:id="@+id/textView9"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Segundo:"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <EditText
        android:id="@+id/editText6_"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="numberDecimal" />

</LinearLayout>

<LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:orientation="horizontal" >

        <Space
            android:layout_width="wrap_content"
            android:layout_height="match_parent" />

        <TextView
            android:id="@+id/textView10"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Longitude (UTC) [-12 a 12]:"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <EditText
            android:id="@+id/editText7_"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="numberSigned" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="70dp"
        android:orientation="horizontal" >

        <RelativeLayout
            android:layout_width="wrap_content"
            android:layout_height="match_parent" >

            <Button
                android:id="@+id/button2"
                android:layout_width="154dp"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:layout_alignParentTop="true"
                android:text="Limpa" />

        </RelativeLayout>

        <Button
            android:id="@+id/button1"
            android:layout_width="146dp"
            android:layout_height="wrap_content"
            android:text="Calcula" />

    </LinearLayout>

    <TextView
        android:id="@+id/textView13"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/textView14"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/textView15"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/textView12"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```

4.5.4 Programação do Aplicativo (Controller)

```
package com.example.projintegrador;

import android.R.string;
import android.os.Bundle;
import android.view.View;
import android.app.Activity;
import android.widget.Button;
import android.widget.TextView;
import android.widget.EditText;
import java.text.DecimalFormat;
import android.view.View.OnClickListener;

public class InsolacaoTerrestreActivity extends Activity {
	boolean fCalcula;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_insolacao_terrestre);

		Button button1 = (Button) findViewById(R.id.button1);
		Button button2 = (Button) findViewById(R.id.button2);

		button1.setOnClickListener(new OnClickListener () {

			public void onClick (View arg0) {

			    try {
			       execInsolacoTerrestre ();
			       } catch (Exception e) {
					// TODO: handle exception
					Button btn = (Button)
findViewById(R.id.button1);
					btn.setError("Por favor, preencha os campos
corretamente e obrigado");
			       }
			}
		}
		);

		button2.setOnClickListener(new OnClickListener () {

			public void onClick (View arg0) {

				Limpa ();
			}
		}
		);
	}

	private void Limpa () {
		EditText editText1 = (EditText) findViewById(R.id.editText1);
		EditText editText3 = (EditText) findViewById(R.id.editText3);
		EditText editText4 = (EditText) findViewById(R.id.editText4_);
		EditText editText5 = (EditText) findViewById(R.id.editText5_);
		EditText editText6 = (EditText) findViewById(R.id.editText6_);
		EditText editText7 = (EditText) findViewById(R.id.editText7_);
		TextView editText13 = (TextView) findViewById(R.id.textView13);
		TextView editText14 = (TextView) findViewById(R.id.textView14);
		TextView editText15 = (TextView) findViewById(R.id.textView15);
		TextView editText12 = (TextView) findViewById(R.id.textView12);

		editText1.setText("");
		editText3.setText("");
		editText4.setText("");
		editText5.setText("");
		editText6.setText("");
		editText7.setText("");
		editText13.setText("");
		editText14.setText("");
		editText15.setText("");
		editText12.setText("");
		LimpaFimMes ();
		editText1.setFocusable(true);
```

```
            editText1.requestFocus();
    }

    private void LimpaFimMes () {
            EditText editText2 = (EditText) findViewById(R.id.editText2);
            TextView textView2 = (TextView) findViewById(R.id.textView2);

            editText2.setText("");
            textView2.setText("Dia [1 a 31*]:");
    }

    // Método Ajustar UTC
private String utcToStr (double timeUTC) {
       String utcStr;

    // Ajustar timeUTC
    if (Double.compare(timeUTC, 0.0)  >= 0) {
        utcStr = String.valueOf(timeUTC);

        if (utcStr.length() == 1) {
            utcStr = "+0" + utcStr;
        } else {
            utcStr = "+" + utcStr;
        }
    } else {
        utcStr = String.valueOf(Math.abs(timeUTC));

        if (utcStr.length() == 1) {
            utcStr = "-0" + utcStr;
        } else {
            utcStr = "-" + utcStr;
        }
    }

    return utcStr;
}

// Método verifica se o ano é bissesto
private boolean bissesto (int a) {
    boolean bissesto; //bi: logico
    int ac; //: inteiro

    if (a < 1){
       ac = a-1;
    }else{
       ac = a;
    }

    if ((((ac % 4) == 0) && ((ac % 100) != 0)) || ((ac % 400) == 0)) {
       bissesto = true;
    } else {
       bissesto = false;
    }

    return bissesto;
}

//Método converte o número referente ao mês em texto
private String mesToStr (int m) {
    String str = "";

    switch (m){
           case 1:
               str = "janeiro";
                      break;
           case 2:
                  str = "fevereiro";
                  break;
           case 3:
                  str = "março";
                  break;
           case 4:
                  str = "abril";
                  break;
           case 5:
               str = "maio";
                  break;
```

```
            case 6:
             str = "junho";
                break;
            case 7:
             str = "julho";
                break;
            case 8:
             str = "agosto";
                break;
            case 9:
                str = "setembro";
                break;
         case 10:
                str = "outubro";
                break;
         case 11:
                str = "novembro";
                break;
         case 12:
                str = "dezembro";
                break;
   }

   return str;
}

// Método que calcula o numero de dias corridos no ano
private int diaDoAno(int d, int m, boolean bi){
   int dias;
   int diasAno[] = new int[12];

   diasAno[0]  = 0;
   diasAno[1]  = 31;
   diasAno[2]  = 59;
   diasAno[3]  = 90;
   diasAno[4]  = 120;
   diasAno[5]  = 151;
   diasAno[6]  = 181;
   diasAno[7]  = 212;
   diasAno[8]  = 243;
   diasAno[9]  = 273;
   diasAno[10] = 304;
   diasAno[11] = 334;

   if ((bi == true) && (m > 2)) {
      dias = diasAno[m-1] + d + 1;
      } else {
         dias = diasAno[m-1] + d;
      }

   return dias;
}

// Método que informa quantos dias tem um determinado mês
private int diasMes(int ms, boolean bi) {
   int dias;
   int diasDoMes[] = new int[12];

   diasDoMes[0]  = 31;
   diasDoMes[1]  = 28;
   diasDoMes[2]  = 31;
   diasDoMes[3]  = 30;
   diasDoMes[4]  = 31;
   diasDoMes[5]  = 30;
   diasDoMes[6]  = 31;
   diasDoMes[7]  = 31;
   diasDoMes[8]  = 30;
   diasDoMes[9]  = 31;
   diasDoMes[10] = 30;
   diasDoMes[11] = 31;

   if ((bi == true) && (ms == 2)) {
      dias = diasDoMes[ms-1] + 1;
   } else {
      dias = diasDoMes[ms-1];
   }
```

```
       return dias;
    }

    // Método que converte grau, minuto e segundo em decimal
    private double gmsToDec(double g, double m, double s) {
       return g+m/60+(s/60)/60;
    }

    // Método que converte a hora parrasda em decimal para hora, minuto e segundo
    private String timeToStr (double time) {
       String strHora, strMinuto, strSegundo;

       strHora = String.valueOf((int) time) + "h ";
       strMinuto = String.valueOf((int)((time-((int)time))*60)) + "min ";
       strSegundo = String.format("%.2f",((time-(int)time)*60-((int)((time-
(int)time)*60)))*60)+"s";

       return strHora + strMinuto + strSegundo;
    }

    // Método que plota o título do app na interface
    private void head(){

        //System.out.println("================================================
========");
         //System.out.println("                  INSOLAÇÂO TERRESTRE");

//System.out.println("============================================================
");
    }

    // Método que plota o resultado para o usuário
    private void saida(int d, String m, int a, double hi, double ha, double hp,
String sUTC) {
              final TextView editText13 = (TextView)
findViewById(R.id.textView13);
              final TextView editText14 = (TextView)
findViewById(R.id.textView14);
              final TextView editText15 = (TextView)
findViewById(R.id.textView15);
              final TextView editText12 = (TextView)
findViewById(R.id.textView12);

           editText13.setText(String.valueOf("INSOLAÇÃO TERRESTRE EM " +
String.valueOf(d)+ " DE " + m.toUpperCase()+" DE "+String.valueOf(a)+":"));
           editText14.setText(String.valueOf("Duração prevista de " +
timeToStr(hi) + " " + sUTC + " UTC;"));
           editText15.setText(String.valueOf("Aurora prevista às " +
timeToStr(ha) + " " + sUTC + " UTC;"));
           editText12.setText(String.valueOf("Poente previsto às " +
timeToStr(hp) + " " + sUTC + " UTC;"));
    }

    private int anoInput () {
       int ano_;
        EditText editText1 = (EditText) findViewById(R.id.editText1);

           try{
               ano_ = Integer.parseInt(editText1.getText().toString()); // 2016

                  if ((ano_ < -2418) || (ano_ > 5582)) {
                     editText1.setError("O ano está fora da faixa!");
                     fCalcula = false;
                  }
           }catch (Exception e){
              editText1.setText("");
              editText1.setError("Preencha o ano, obrigado!");
              LimpaFimMes();
              fCalcula = false;
              ano_ = 0;
           }

           return ano_;
    }

    private int mesInput (boolean fBissesto) {
       int mes_;
```

```
        EditText editText2 = (EditText) findViewById(R.id.editText2);

        try{
                    mes_ = Integer.parseInt(editText2.getText().toString()); //
2016

                    if ((mes_ < 1) || (mes_ > 12)) {
                        editText2.setError("O mês inserido não existe!");
                        fCalcula = false;
                    }else{
                        // TODO: faixaDia("Dia [1 a
"+Integer.toString(diasMes(mes_, fBissesto_))+"]: ");
                        AlteraFimMes(diasMes(mes_, fBissesto));
                    }
        }catch (Exception e){
                mes_=0;
                LimpaFimMes();
                editText2.setError("Insira um mês, obrigado!");
                fCalcula = false;
        }

            return mes_;
    }

    private void AlteraFimMes(int dia) {
        final TextView editText20 = (TextView) findViewById(R.id.textView2);

        switch (dia) {
                case 28:
                        editText20.setText("Dia [1 a 28*]:");
                        break;
                case 29:
                        editText20.setText("Dia [1 a 29*]:");
                        break;
                case 30:
                        editText20.setText("Dia [1 a 30*]:");
                        break;
                case 31:
                        editText20.setText("Dia [1 a 31*]:");
                        break;
                default:
                        editText20.setText("Dia [1 a 31*]:");
                        break;
                }
    }

    private int diaInput(int mes_, boolean fBissesto_){
        int dia_;
        EditText editText3 = (EditText) findViewById(R.id.editText3);

        try{
                    dia_ = Integer.parseInt(editText3.getText().toString()); //
2016

                    if ((dia_ < 1) || (dia_ > diasMes(mes_, fBissesto_))) {
                        editText3.setError("Corrija o dia por favor!");
                        fCalcula = false;
                    }
        }catch (Exception e){
                dia_= 0;
                editText3.setText("");
                editText3.setError("Digite o dia por favor!");
                fCalcula = false;
        }

            return dia_;
    }

    private double latitudeGrausInput () {
        double latitudeGraus_;
            EditText editText4 = (EditText) findViewById(R.id.editText4_);

            try{
                    latitudeGraus_ =
Double.parseDouble(editText4.getText().toString());

                    if ((latitudeGraus_ < -90) || (latitudeGraus_ > 90)) {
```

```
                    editText4.setError("Corrija o grau da latitude por
favor!");
                    fCalcula = false;
                }
            }catch (Exception e){
                latitudeGraus_=0;
                editText4.setText("");
                editText4.setError("Digite o grau da latitude por favor!");
                fCalcula = false;
            }

            return latitudeGraus_;
    }

    private double latitudeMinutosInput () {
        double latitudeMinutos_;
            EditText editText5 = (EditText) findViewById(R.id.editText5_);

        try{
                    latitudeMinutos_ =
Double.parseDouble(editText5.getText().toString());

                    if ((latitudeMinutos_ < 0) || (latitudeMinutos_ >= 60)) {
                        editText5.setError("Corrija o minuto da latitude por
favor!");
                        fCalcula = false;
                    }
        }catch (Exception e){
                latitudeMinutos_=0;
                editText5.setText("");
                editText5.setError("Digite o minuto da latitude por favor!");
                fCalcula = false;
        }

            return latitudeMinutos_;
    }

    private double latitudeSegundosInput(double latitudeGraus_, double
latitudeMinutos_) {
        double latitudeSegundos_;
        EditText editText6 = (EditText) findViewById(R.id.editText6_);

        try{
                    latitudeSegundos_ =
Double.parseDouble(editText6.getText().toString());

                    if (((latitudeSegundos_ < 0) || (latitudeSegundos_ >= 60)) ||
((latitudeGraus_ == 0) &&
                    (latitudeMinutos_ == 0) && (latitudeSegundos_ == 0))) {
                        editText6.setError("Corrija o segundo da latitude por
favor!");
                        fCalcula = false;
                    }
        }catch (Exception e){
                latitudeSegundos_=0;
                editText6.setText("");
                editText6.setError("Digite o segundo da latitude por favor!");
                fCalcula = false;
        }

            return latitudeSegundos_;
    }

        private int UTCinput () {
                int UTC_;
                EditText editText7 = (EditText) findViewById(R.id.editText7_);

                try{
                    UTC_ = Integer.parseInt(editText7.getText().toString()); //
2016

                    if ((UTC_ < -12) || (UTC_ > 12)) {
                        editText7.setError("Corrija o valor UTC do Fuso Horário por
favor!");
                        fCalcula = false;
                    }
                }catch (Exception e){
```

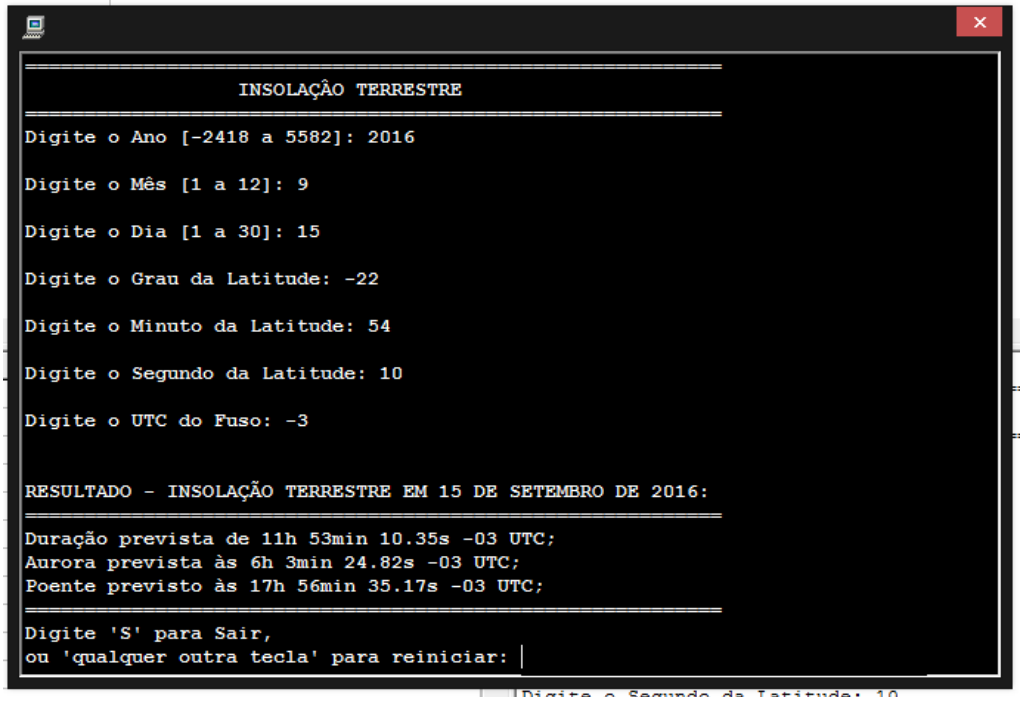
```
                    UTC_=0;
                    editText7.setText("");
                    editText7.setError("Insira um valor UTC para o Fuso Horário
por favor!");
                    fCalcula = false;
                }

            return UTC_;
    }

    // Método de controle lógico do aplicativo
    private void execInsolacoTerrestre () {
       // Variáveis de Valores de Entrada
       int dia, mes, ano;
       double latitudeGraus, latitudeMinutos, latitudeSegundos, UTC;

       // Variáveis de valores internos
       double grausEquadorSolar, minutosEquadorSolar, segundosEquadorSolar;
       double equinocio;
       String mesStr, utcStr;

       // Variáveis de Valores Calculados Auxiliares
       boolean fBissesto;
       int diasAnoCivil, diasX;
       double equadorSolar, latitude;

       // Variáveis de Valores Calculados de Saída
       double horasDeInsolacao, HoraAurora, HoraPoente;

       // Comandos

       // Parâmetros Iniciais
       grausEquadorSolar = 23;
       minutosEquadorSolar = 27;
       segundosEquadorSolar = 0;
       equinocio = 284.5;

       dia = 0;
       mes = 0;
       ano = 0;

       latitudeGraus = 0;
       latitudeMinutos = 0;
       latitudeSegundos = 0;
       UTC = 0;

       mesStr = "";
       utcStr = "";

       fBissesto = false;
       diasAnoCivil = 0;
       diasX = 0;

       equadorSolar = 0;
       latitude = 0;

       horasDeInsolacao = 0;
       HoraAurora = 0;
       HoraPoente = 0;

       // Dados de Entrada do Usuário
       //head();
       fCalcula = true;

       ano = anoInput();
       fBissesto = bissesto(ano);
       mes = mesInput(fBissesto);
       dia = diaInput(mes, fBissesto);
       latitudeGraus = latitudeGrausInput();
       latitudeMinutos = latitudeMinutosInput();
       latitudeSegundos = latitudeSegundosInput(latitudeGraus, latitudeMinutos);
       UTC = UTCinput();

       // Indicação de hora local em relação a hora GMT, DST não está previsto
       utcStr = utcToStr(UTC);

       if (fCalcula == true){
```

```
                // Ano Civil
                if (fBissesto = true){
                   diasAnoCivil = 366;
                   } else {
                         diasAnoCivil = 365;
                   }

                // Ajustar o Mês por Extenso
                mesStr = mesToStr(mes);

                // Conta dias no ano
                diasX = diaDoAno(dia, mes, fBissesto);

                // Concatenar e Ajustar Valores
                equadorSolar = gmsToDec(grausEquadorSolar, minutosEquadorSolar,
segundosEquadorSolar);
                latitude = gmsToDec(latitudeGraus, latitudeMinutos,
latitudeSegundos);

                // Ajuste da latitude para evitar error de divisão por zero na
função
                if (latitude == 0) {
                   latitude = 0.000000001; // Valor suficiente para ser
insignificante na ordem de grandeza nas dimenções para latitude terrestre
                }

                // Cálculos de Insolação, Auróra e Poente
                horasDeInsolacao =  ((2.0 / 15.0) * Math.acos(-Math.tan(Math.PI /
180.0 * latitude) *
                           Math.tan(Math.PI / 180.0 * equadorSolar * Math.sin(Math.PI
/ 180.0 * (360.0 /
                           diasAnoCivil * (equinocio + diasX))))) * 180.0 / Math.PI);

                HoraAurora = 12.0 - ((2.0 / 15.0) * Math.acos(-Math.tan(Math.PI /
180.0 * latitude) *
                           Math.tan(Math.PI / 180.0 * equadorSolar * Math.sin(Math.PI
/ 180.0 * (360.0 /
                           diasAnoCivil * (equinocio + diasX))))) * 180.0 / Math.PI)
/ 2.0;

                HoraPoente = 12.0 + ((2.0 / 15.0) * Math.acos(-Math.tan(Math.PI /
180.0 * latitude) *
                           Math.tan(Math.PI / 180.0 * equadorSolar * Math.sin(Math.PI
/ 180.0 * (360.0 /
                           diasAnoCivil * (equinocio + diasX))))) * 180.0 / Math.PI)
/ 2.0;

                // Saida
                saida(dia, mesStr, ano, horasDeInsolacao, HoraAurora, HoraPoente,
utcStr);
          }
      }
}
```

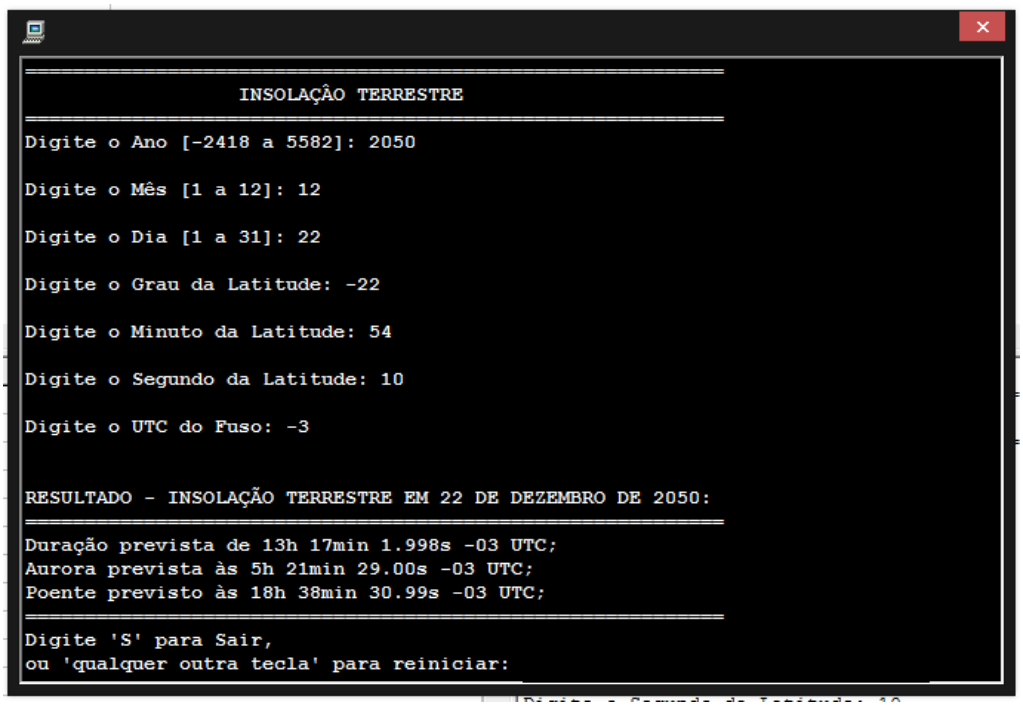
4.6 Testes.

4.6.1 Testes no Visualg

1º Teste – 15 de Setembro de 2016 em -22º54’10’’ S -3 UTC

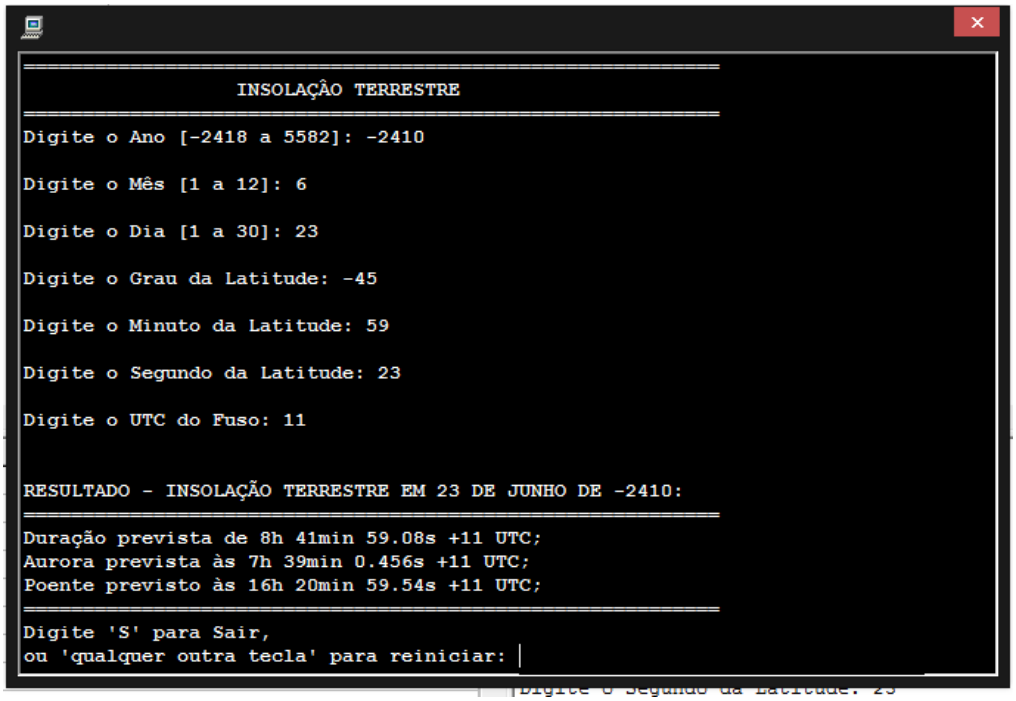


2º Teste – Dia 22/12/2050 Lat -22° 54’ 10’’ -03 UTC

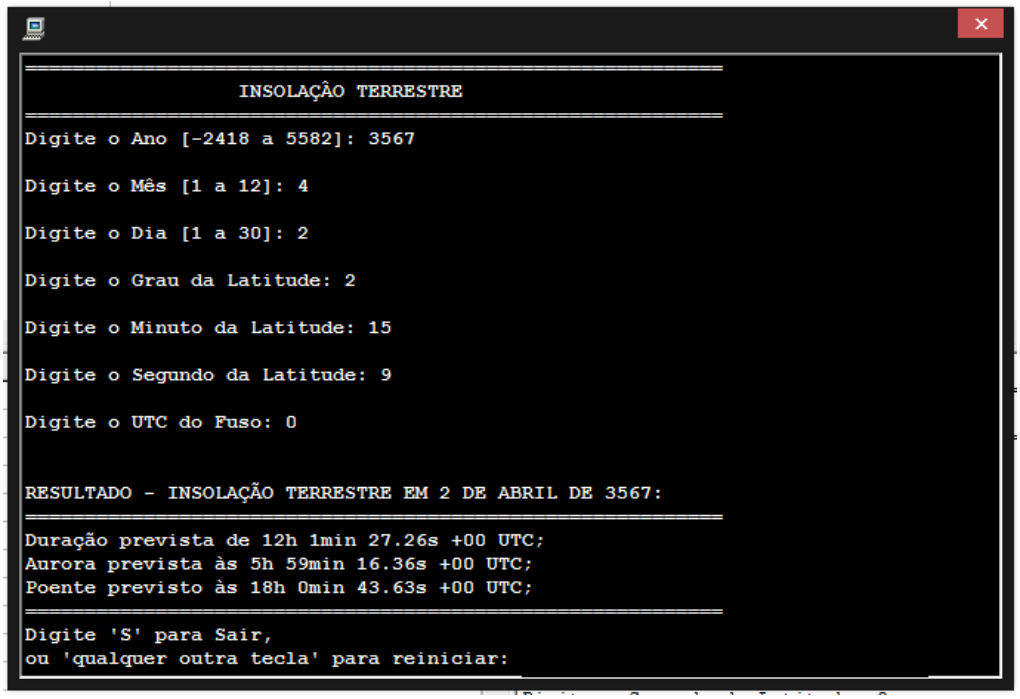


3º Teste – Dia 22/03/0300 lat 15° 56’ 45’’ 05 UTC

4º Teste – Dia 23/06/2410 a.C lat -45° 59’ 23’’ 11 UTC



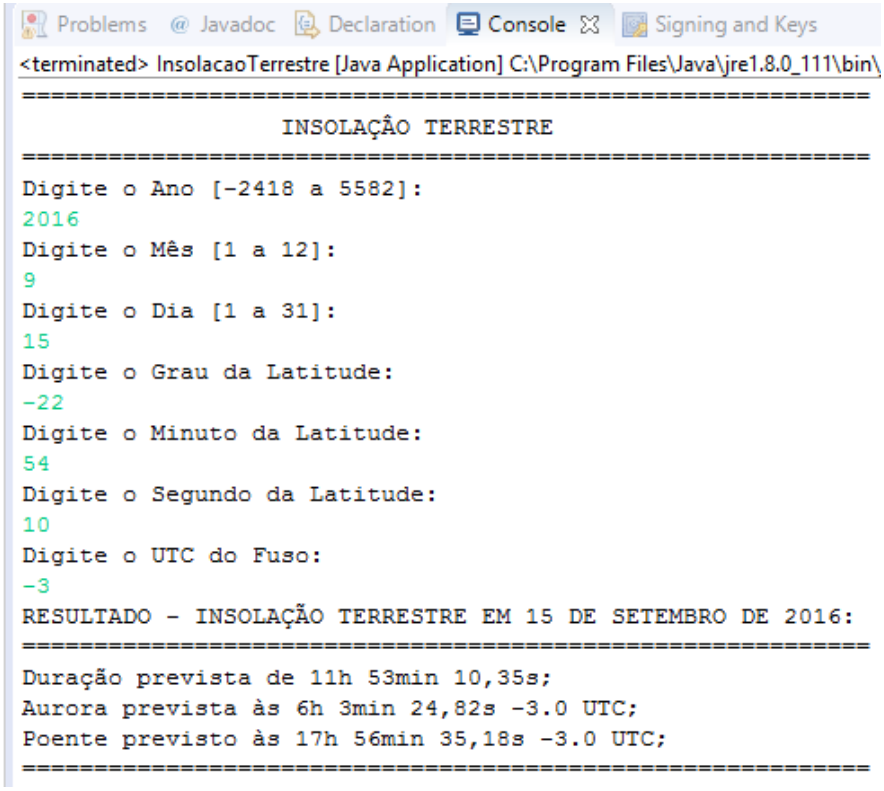
5º Teste – Dia 02/04/3567 lat 2° 15’ 9’’ 0 UTC



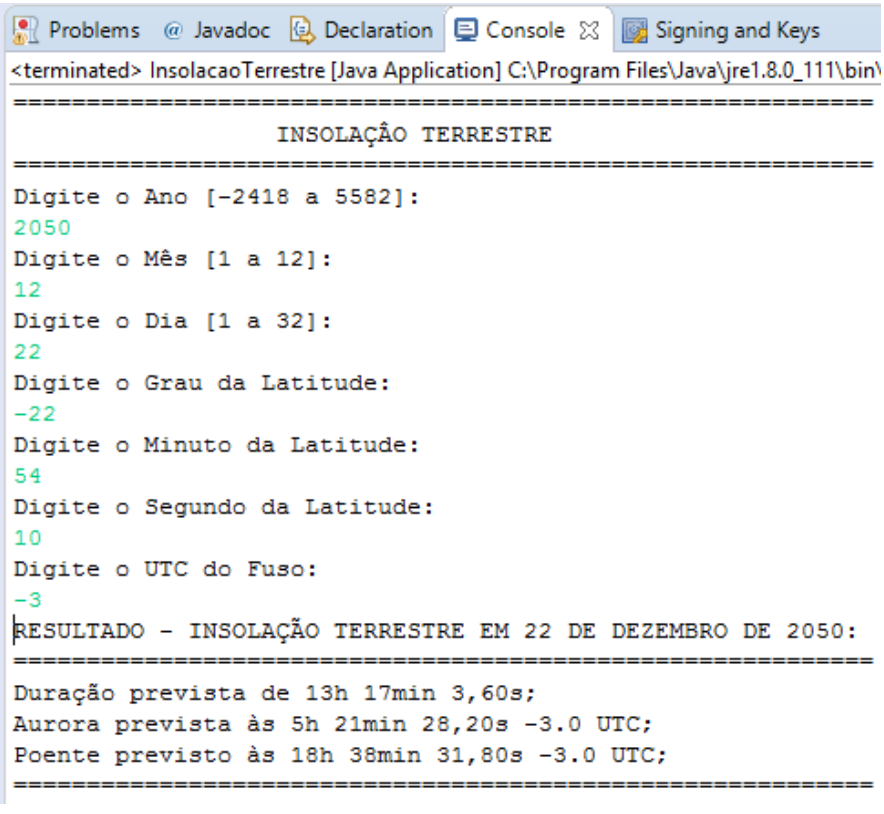
6º Teste – Dia 01/01/2017 lat -22° 54’ 10’’ -03 UTC

4.6.2 Testes no Console Java

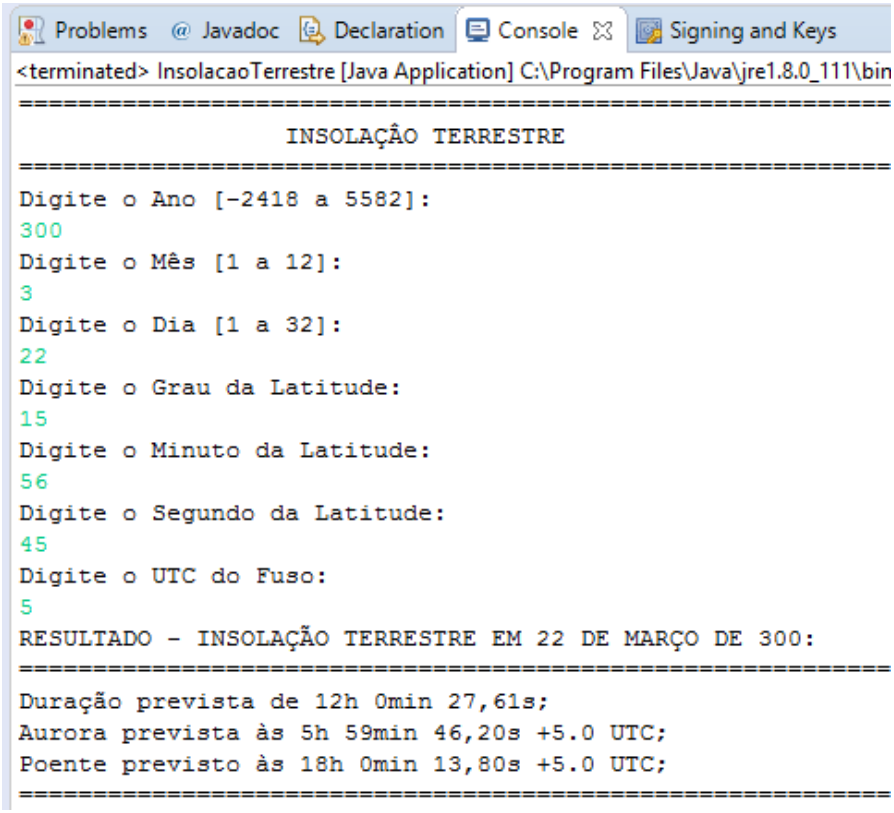
1º Teste – 15 de Setembro de 2016 em -22º54’10’’ S -3 UTC



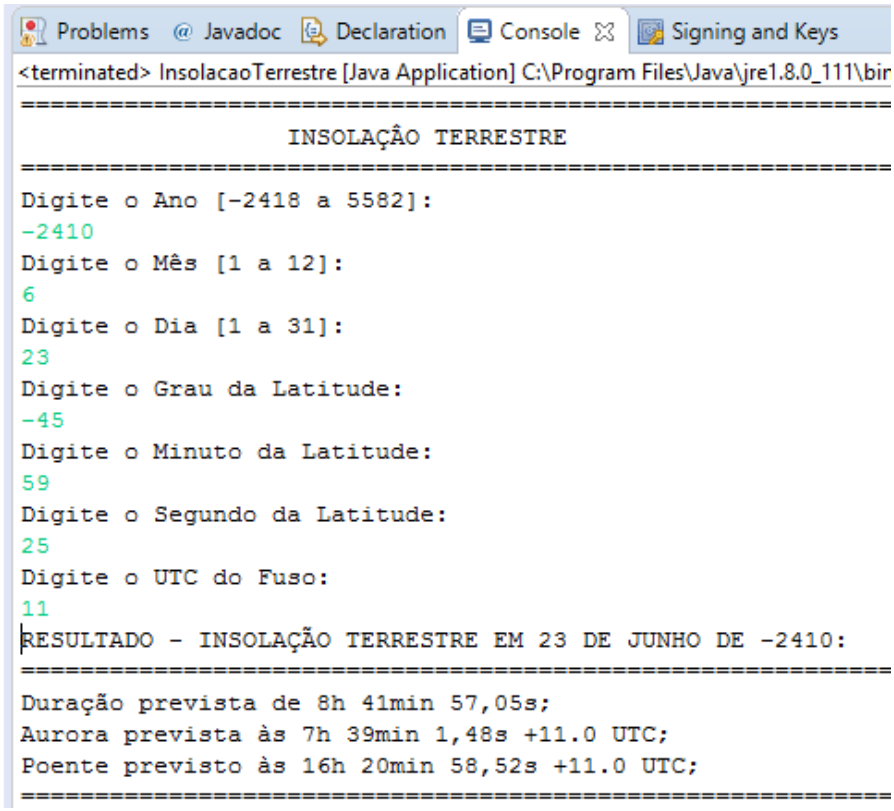
2º Teste – Dia 22/12/2050 Lat -22° 54’ 10’’ -03 UTC



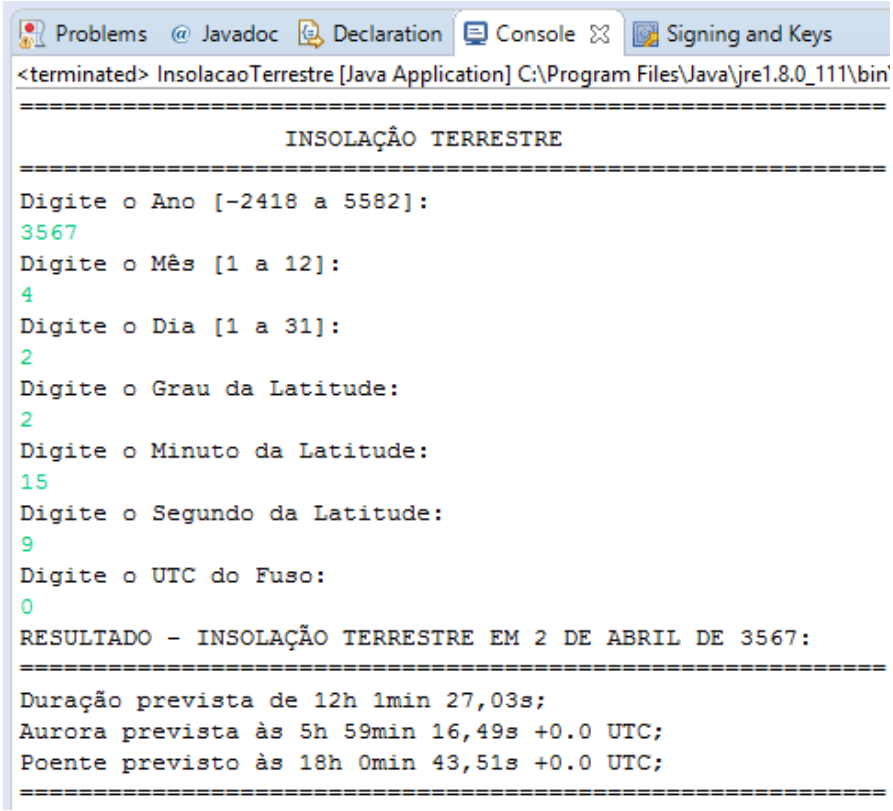
3º Teste – Dia 22/03/0300 lat 15° 56’ 45’’ 05 UTC



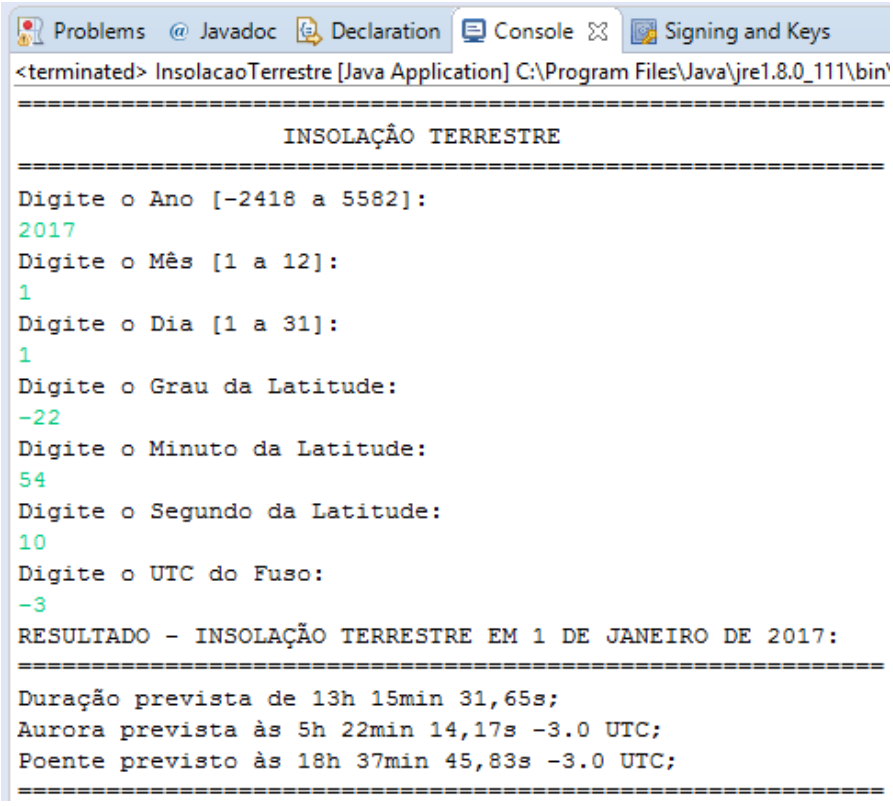
4º Teste – Dia 23/06/2410 a.C lat -45° 59’ 23’’ 11 UTC



5º Teste – Dia 02/04/3567 lat 2° 15’ 9’’ 0 UTC

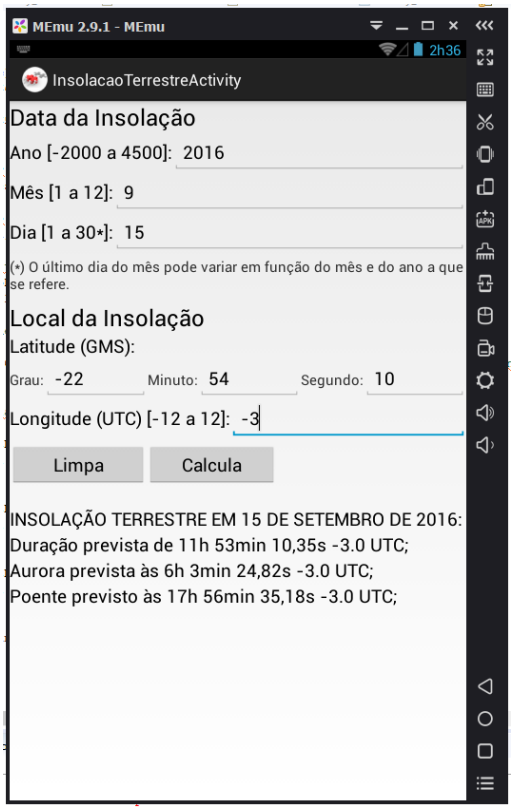


6º Teste – Dia 01/01/2017 lat -22° 54’ 10’’ -03 UTC

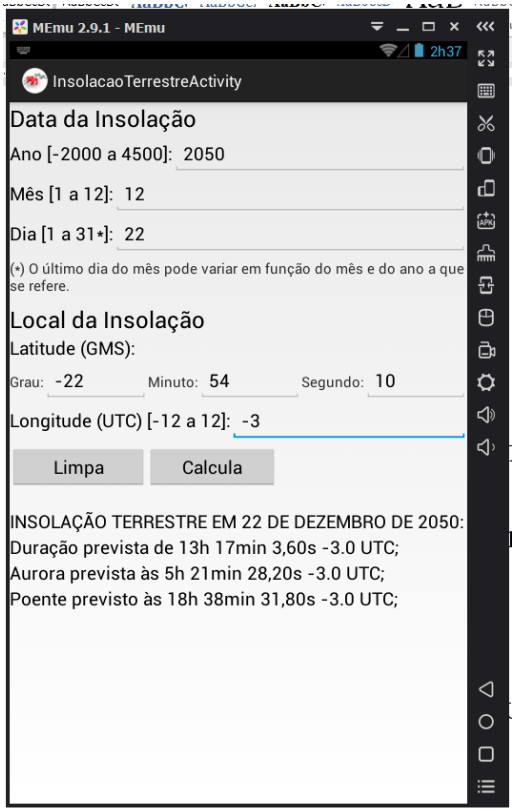


4.6.1 Testes no Aplicativo Android

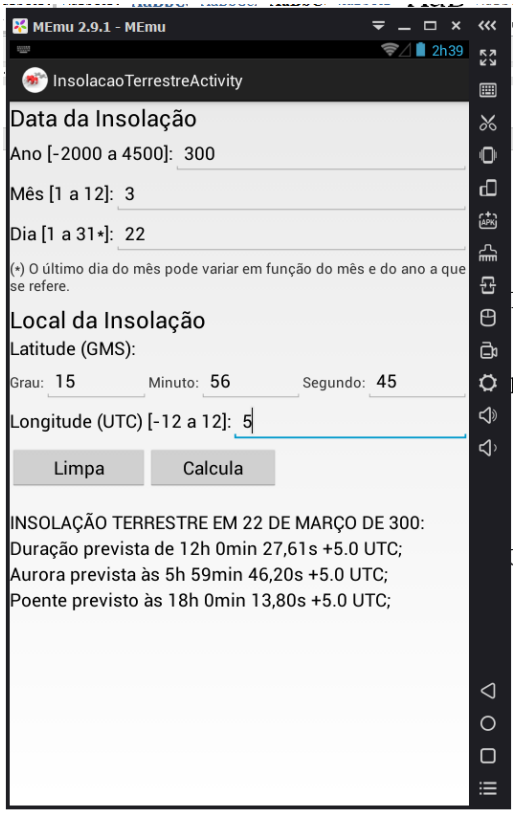
1º Teste – 15 de Setembro de 2016 em -22º54’10’’ S -3 UTC



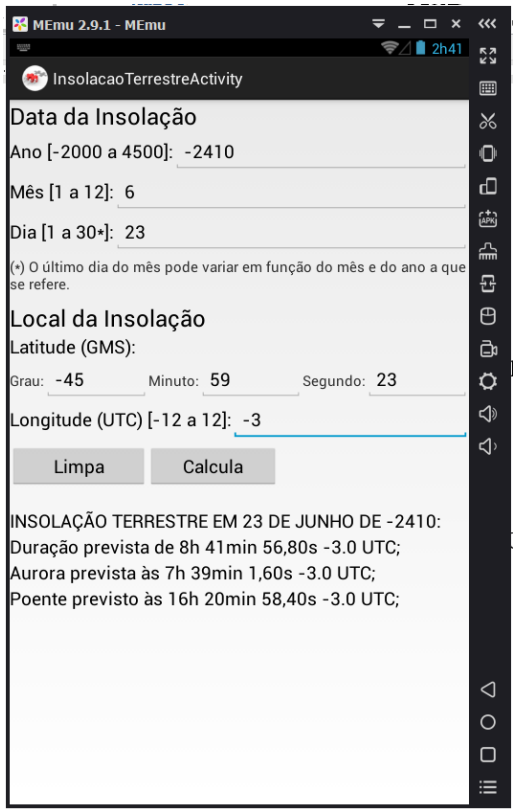
2º Teste – Dia 22/12/2050 Lat -22° 54’ 10’’ -03 UTC



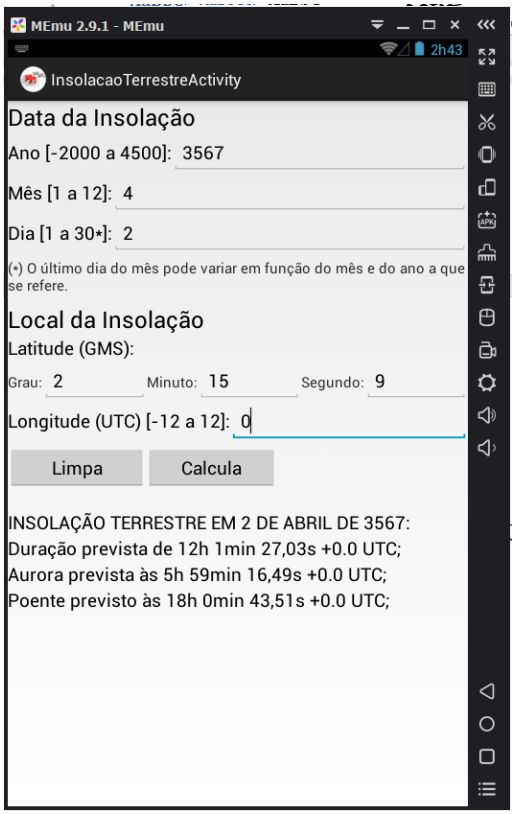
3º Teste – Dia 22/03/0300 lat 15° 56’ 45’’ 05 UTC



4º Teste – Dia 23/06/2410 a.C lat -45° 59’ 23’’ 11 UTC



5º Teste – Dia 02/04/3567 lat 2° 15’ 9’’ 0 UTC



6º Teste – Dia 01/01/2017 lat -22° 54’ 10’’ -03 UTC

CAPÍTULO 7 – Bibligrafia

- Portal SBF (Sociedade Brasileira de Física), O Calendário Gregoriano - disponível em: http://www.sbfisica.org.br/rbef/pdf/vol17a06.pdf Acesso em 16 de Setembro de 2016.

- Portal da UFES - Centro de ciências Exatas, Origem da Evoluão do Nosso Calendário - disponível em: http://www.cce.ufes.br/jair/ieff/Origem_evolu%C3%A7%C3%A3o_calend%C3%A1rio_adaptado_Brasil.pdf> Acesso em 16 de Setembro de 2016.

- Portal UFMG – Instituto de Ciências Exatas, Calendários - disponível em: http://www.mat.ufmg.br/~espec/Monografias_Noturna/Monografia_CristianaCarmo.pdf Acesso em 16 de Setembro de 2016.

- Portal do Departamento de Astronomia do Instituto de Física da UFES, Presseção do Eico da Terra - disponível em: http://astro.if.ufrgs.br/fordif/node8.htm Acesso em 16 de Setembro de 2016

- Portal do Instituto de Física da UFRJ, Movimento de Nutação - disponível em: http://www.if.ufrj.br/~marta/caronte-hipertexto/movimentosdaterra/portugues_nutacao_descricao.html Acesso em 18 de Setenbro de 2016.