COP 3502
Project 1
Due September 20, 2015 (11:59pm)

Background

There is a "pay to play" cell phone game in which users utilize augmented reality to capture various monsters, called "*Brokemon-Go*". In the game, players pay to capture various monsters that are in their environment. If a monster is captured, the player receives money. If the monster is not captured, the player loses their money. The game application tells the player the monster's location (in x, y coordinates), its time of appearance (T_a) and the duration of its existence (T_e). This means that the monster will appear at time T_a and will exist from time T_a to $T_a + T_e$. The objective of the game is for the user to get to the location of the monster that it is trying to capture, before the monster disappears.

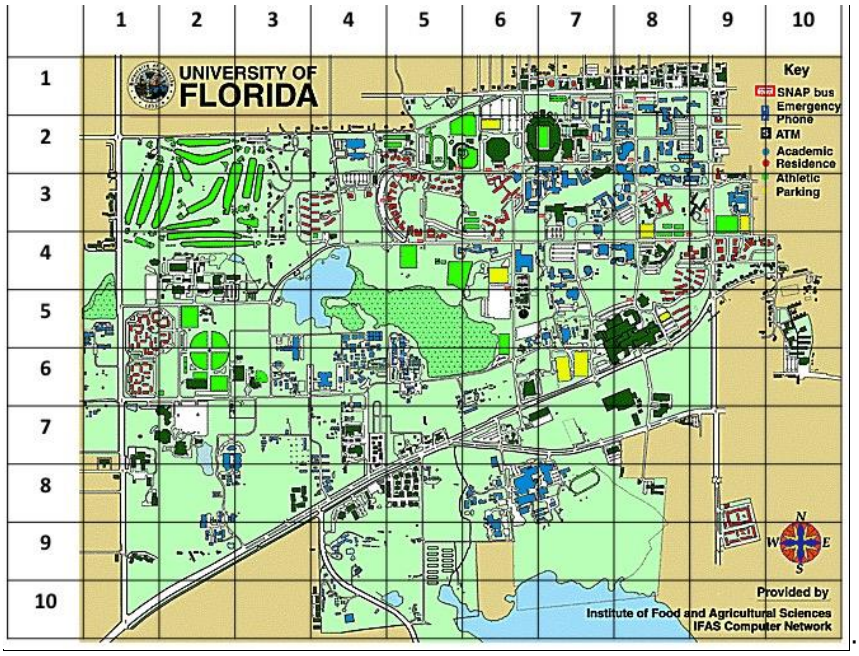
One of your friends is severely addicted to this game. Your friend is about to go broke playing the game because the monster disappears right before they are able to make it to the monster's location. Your friend just found out that you are learning how to program and has asked you to help them develop some code to help them play the game better, and not lose any more of their money. Your friend would like you to help them decide or **select** if it would be wise to pursue a given monster. Please see the specific rules in the Section "Probability of Capture".

You have agreed to help your friend. After speaking with the game company, you have found out that they have created a new rule. The rule is: If a player's playerID ends in 00 to 49, this person is on the "lucky list"; however if the playID ends in 50 to 99, this person is on the "normal list". People on the lucky list are given special privileges, which are outlined in the "Probability of Capture" section.

Your Assignment

Create a piece of code that takes in the monster's location, the monster's appear time, the monster's exist time, the player's ID, and the player's location, the player's walking speed, and the player's notice time. After this is done, your program should output the player's likelihood of capturing the monster. The pieces of data that will be supplied at the console are:

- **Monster Location**: supplied as grid locations (x, y) (1 <= x <= 10, and 1 <= y <= 10) where x is the longitude and y is the latitude (see grid below, scale in km)



- **Monster Appear Time:** 1 to 1440 minutes *(24 hours * 60 minutes in a day)*
- **Monster possible Exist Time:** 10 to 59 minutes
- **Player ID: exactly** 8 digits, none beginning with a zero (integer)
- **Player Location**: supplied as grid locations (x, y) where x is the longitude and y is the latitude
- **Player Notice Time**: the time when player notices the monster appear on his/her cellphone app, on the same day as the monster appear. (1 to 1440).
- **Walk speed:** the player's speed to get to the monster's location. (10 to 200 m/minute).

Probability of Capture

You will use the following rules to decide the possibility of a player capturing the monster, based on the time the player get there. If the time the player gets to the monster's location (T_g) is less than or equal to the sum of the monster appear time (T_a) and the possible exist time (T_e), $T_g <= T_a + T_e$, the player will definitely capture the monster. If the player arrives late, i.e., $T_g > T_a + T_e$, the monster's disappear probability increases with the proportion $((T_g - (T_a + T_e))/ T_e) * 100$. The likelihood of capturing the monster is defined by the thresholds below.

Definitely

- The time the player gets there (T_g) is less than or equal to the sum of the monster's appear time (T_a) and the possible exist time (T_e), i.e., $T_g <= T_a+T_e$

Highly Likely

- $T_g > T_a + T_e$
- **Lucky List:** $(T_g - (T_a + T_e)) / T_e$ is within 0 to 10% (inclusive)
- **Normal List:** $(T_g - (T_a + T_e)) / T_e$ is within 0 to 5% (inclusive)

Likely

- $T_g > T_a + T_e$
- **Lucky List:** $(T_g - (T_a + T_e)) / T_e$ is greater than 10%, but within 30% (inclusive)
- **Normal List:** $(T_g - (T_a + T_e)) / T_e$ is greater than 5%, but within 20% (inclusive)

Unsure

- $T_g > T_a + T_e$

- **Lucky List:** $(T_g - (T_a + T_e)) / T_e$ is greater than 30%, but within 40% (inclusive)
- **Normal List:** $(T_g - (T_a + T_e)) / T_e$ is greater than 20%, but within 35% (inclusive)

Unlikely

- $T_g > T_a + T_e$
- **Lucky List:** $(T_g - (T_a + T_e)) / T_e$ is greater than 40%, but within 50% (inclusive)
- **Normal List:** $(T_g - (T_a + T_e)) / T_e$ is greater than 35%, but within 40% (inclusive)

Highly Unlikely

- $T_g > T_a + T_e$
- **Lucky List:** $(T_g - (T_a + T_e)) / T_e$ is greater than 50%
- **Normal List:** $(T_g - (T_a + T_e)) / T_e$ is greater than 40%

Code Structure

1. Read in the information about the monster from the console (**all integers**)
   - Monster Location, Appear Time and Possible Exist Time

2. Read in the information about the player from the console (**all integers**)
   - Player ID
   - Player Walk Speed
   - Player Find Time and Location

3. Determine the player's state
   - What type of player he/she is ("lucky list" vs. "normal list")
   - When the player will arrive at the monster location
   - Whether he/she is late or not; if they are late, what percentage it is

4. Output the possibility the player capturing the monster (definitely, highly likely, likely, unsure, unlikely, highly unlikely). The possibility word(s) should come directly before the word "possibility", regardless of the grammar

Sample Run 1 - valid input (user input is in red)

```
Hello and welcome to the Monster Capture Possibility Calculator.
Please enter the latitude of the monster (1-10): 5
Please enter the longitude of the monster (1-10): 5
Please enter the time of the monster appear (1-1440): 1310
Please enter the possible time of the monster will exist (10-59): 40
Please enter the player's ID (8 digits): 11119999
Please enter the time of the player noticing monster (1-1440 and greater than
the time the monster appears): 1340
Please enter the latitude of the player (1-10): 5
Please enter the longitude of the player (1-10): 6
Please enter the player's walking speed (10-200): 90
```

Output

```
Player 11119999 who is on the normal list,
noticed the monster at time 1340,
is 1000.0 m away from the monster,
and will arrive at time 1351.1.
The monster's disappear time is about 1350.
So the player will capture this monster with highly likely possibility.
```

Sample Run 2 - valid input (user input is in red)

```
Hello and welcome to the Monster Capture Possibility Calculator.
Please enter the latitude of the monster (1-10): 3
Please enter the longitude of the monster (1-10): 7
Please enter the time of the monster appear (1-1440): 250
```

```
Please enter the possible time of the monster will exist (10-59): 25
Please enter the player's ID (8 digits): 11358734
Please enter the time of the player noticing monster (1-1440 and greater than
the time the monster appears): 270
Please enter the latitude of the player (1-10): 8
Please enter the longitude of the player (1-10): 4
Please enter the player's walking speed (10-200): 87
```

Output

```
Player 11358734 who is on the lucky list,
noticed the monster at time 270,
is 5831.0 m away from the monster,
and will arrive at time 337.0.
The monster's disappear time is about 275.
So the player will capture this monster with highly unlikely possibility.
```

Sample Run 3 - valid input (user input is in red)

```
Hello and welcome to the Monster Capture Possibility Calculator.
Please enter the latitude of the monster (1-10): 3
Please enter the longitude of the monster (1-10): 9
Please enter the time of the monster appear (1-1440): 377
Please enter the possible time of the monster will exist (10-59): 34
Please enter the player's ID (8 digits): 23678968
Please enter the time of the player noticing monster (1-1440 and greater than
the time the monster appears): 390
Please enter the latitude of the player (1-10): 6
Please enter the longitude of the player (1-10): 7
Please enter the player's walking speed (10-200): 180
```

Output

```
Player 23678968 who is on the normal list,
noticed the monster at time 390,
is 3605.6 m away from the monster,
and will arrive at time 410.0.
The monster's disappear time is about 411.
So the player will capture this monster with definitely possibility.
```

Notes

- You may assume that we will only give reasonable input, within the range and bounds of the description and of the correct type (integer).
- "Within" means "up to and including". For example, if the possibility is within 5%, this means **possibility <= 5%**.
- Please use `System.out.println("....");` to print your output in lieu of `System.out.print("....");`
- Please use the Math library to support your calculations. You can import it by adding this line to the top of your code:
    - `import java.lang.Math;`
- The straight line distance (d) is also known as the Euclidean distance and can be calculated using the formula below.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

- **Note that**, since the map is in ***km*** scale, you should change it to ***m*** (that is, multiply by 1000) when you using d to calculate the time the player spend on the road.

- To get the absolute value, you can use `Math.abs()`. Abs works as follows:

  - `int a = -1;`
  - `a = Math.abs(a);  //now a=1`

- · The exact time the player will arrive (T_g) is calculated using the formula below.

$$T_g = T_n + (d / S)$$

- Where T_n denotes the time when the player notices the monster, and S denotes the player's walking speed. **Note that**, if d and S are both **integer** types, you should convert at least one to a **double** type to get the most accurate result.

- The distance and the time the player will arrive must be rounded off to include one digit after the decimal. This can be achieved using `Math.round()`. Please place the following line of code at the beginning of your program (before the Class header) to import the Math library's functionality:

  - `import java.lang.Math;`

  - The following example demonstrates how you would use the Math library to round to one decimal place:

    - `double num = 1.34567;`
    - `double roundedNum = Math.round(num*10)/10.0;`

Submission Requirements

- Name the project "`Project1`"
- Name the class "`CaptureCalculator`"
- Zip the java file (r*ight click the java file > Send to > Compressed (zipped) folder*)
- Name the zip file `project1_ulfid.zip` where uflID is the portion of your ufl email account that comes before the **@ufl.edu** part.
- Submit using Canvas

- If you submit the .java file without zipping, your project will not be graded.
- If you create multiple Scanner variables to read from the keyboard, all of your input may not be received by the grading program
- If your file is not named `project1_ulfid.zip`, your project will not be graded. If you submit multiple times to Canvas, it will append a number to your submission. This is fine. We will still be able to grade it
- If you do not name the class exactly as specified, your project will not be graded.
- It is highly recommended that you test your program piece by piece before assembling your final code. You will have a much easier time if you build your program piece by piece rather than trying to write the entire program. Pay very close attention to the order that you give inputs to the sample

program. Your output **should not differ in any way** from the sample output. Using additional prompts, words, abbreviations, etc may result in the grading program taking unnecessary points off.

- If you want to check and ensure that you have done this correctly, please use the input redirection and diff operations that were covered in class.

Grading

- 90% of your program will be graded on its ability to generate proper output given reasonable inputs without producing errors.
- 10% of your program will be graded on its adherence to the coding style standards.