CS211 – Programming Practicum **Fall 2014**

Programming Project 10

Due: Friday, 10/31/14 at 11:59 pm

AI Guessing Game using Java Objects

For this program, the computer will try to predict whether the user will guess "heads" or "tails". The AI algorithm relies on the fact that while a person thinks they are coming up with random guesses, they are actually following a pattern even if they don't realize it. The AI algorithm will remember user's patterns and use this to predict the user's next guess.

The program/user interaction is as follows:

1. The user needs to pick either heads or tails.
2. Before the user makes his/her pick, the program will make a prediction as to whether the user is going to pick heads or tails.
3. The program's prediction is hidden from the user until he/she makes his/her actual pick.
4. Once the user makes his pick the program's prediction is revealed.
5. If the program correctly predicted the user's pick, the program gets a point. Otherwise, if the program does not correctly predict the user's pick, the user gets a point.
6. Whoever (program or user) gets 25 points first is deemed the winner.

Below is a possible sample of the input/output of the program. The user's guess (either h or t) is shown in bold type.

```
Welcome to MindReader

Guess heads or tails and I'll predict your guess.
What is your guess [h/t] ? t
Oh no. I predicted h. You get a point.
Score = 0 | 1

Guess heads or tails and I'll predict your guess.
What is your guess [h/t] ? h
Yes! I predicted h.  I get a point.
Score = 1 | 1

Guess heads or tails and I'll predict your guess.
What is your guess [h/t] ? h
Oh no. I predicted t.  You get a point.
Score = 1 | 2

Guess heads or tails and I'll predict your guess.
What is your guess [h/t] ?
...
```

An array of java objects will be used to store the data needed for this project. These object will need to store 3 values at each location in the array:

- a 4-character string specifying the last 4 guesses made by the user
- the number of times the user guessed heads after this specific pattern of 4 guesses
- the number of times the user guessed tails after this specific pattern of 4 guesses

There are 16 possible patterns that the last 4 guesses could generate, so the array will need 16 positions holding 16 different objects.

The idea behind the AI algorithm is that a person is likely repeat a 5th guess after a specific pattern of 4 guesses. The computer will keep track of the user's last four guesses, look up this pattern in the array and use the information stored to predict whether the user's next guess will be heads or tails. If the data stored in the array regarding the last four guesses indicates more heads than tails have been guessed by the user, the computer will predict heads. If the data stored indicates more tails than heads have been guess, the computer will predict tails. If the user has made the same number of heads and tails or if this pattern is not contained in the hash table, the computer will make a random prediction.

For the AI algorithm to work, there must be enough guesses made to establish some pattern. This is why the game is being played to 25 points. If played to a smaller number of points, the computer will pretty much be making random predictions.

The high level structure of this program is:

1. The computer makes a prediction based on the pattern of the last 4 user guesses from the information stored in the array and conceals it from the user.
2. The user makes a choice/guess and enters it via standard input.
3. The computer updates the score appropriately as to whether it made a correct prediction or not.
4. The computer stores the user's choice/guess in the array entry for the pattern of the previous 4 user guesses by incrementing the number of times heads or tails was guessed for that pattern.
5. If 25 points has not been reached by the computer or user, go to step 1 updating the pattern with the most recent guess. If 25 points has been reached, print a message declaring who won and quit the program.

This program will require a translation from the pattern for the last 4 guesses to the position in the array. Since there are 16 possible patterns for the last 4 guesses, we could encode a guess of similar to binary numbers. A guess of "heads" could translate to a digit of 1. While a guess of "tails could translate to a digit of 0. The translation between pattern of the last 4 guesses and the position of the array could be as follows:

Guess Pattern	Binary Translation	Array Position
tttt	0000	0
ttth	0001	1
ttht	0010	2
tthh	0011	3
thtt	0100	4
thth	0101	5
thht	0110	6
thhh	0111	7

Guess Pattern	Binary Translation	Array Position
httt	1000	8
htth	1001	9
htht	1010	10
hthh	1011	11
hhtt	1100	12
hhth	1101	13
hhht	1110	14
hhhh	1111	15

Let us assume the user made the following guesses:

1. heads
2. heads
3. tails
4. heads

The four character string created by this pattern would be: hhth
Which, according to the above table, would translate to array position 13.

The computer algorithm would look at array position 13 to determine whether it should predict heads or tails. If the object in position 13 has a higher “head count” than the “tail count”, it will predict heads. If the object has a higher “tail count” than the “head count”, it will predict tails. If the “head count” is the same as the “tail count”, use the random number generator to determine whether to predict heads of tails.

Now let us assume the user’s next guess is tails. In this case, the item in the array with pattern of hhth (position 13) will increment by 1 the “tail count” for this pattern.

Now the four character string based on the last four guesses would be: htht
Which, according to the above table, would translate to array position 10.

Also note that for the first four guesses, there have not been enough guesses made to build a four character string of previous guesses. In this case, we could either just use a random number generator to determine whether the computer picks heads or tails or assume that we start the program will some predetermined 4 guesses (such as all tails or all heads or some other pattern you like).

Displaying the Values Stored in the Array

Your program is to display the entire contents array at the end of the game when 25 points has been reached and whenever the user enters a value of “d” when being asked for a guess of heads or tails. Thus at the prompt of:

```
What is your guess [h/t] ?
```

your program has four options to check for:

1. a value of “h” for heads,
2. a value of “t” for tails,
3. a value of “d” to display the entire hash table,
4. or some other value that should result in an error message.

When displaying the contents of the array, you must list the position in the array where the item is stored along with the pattern associated with that position and the number of heads and tails guessed with that pattern.

JAVA OBJECTS/CLASSES

A Java class is an improvement on the C struct. C structs just allow the program to group some data elements together. While a Java class allows the program to group data elements together along with the operations needed to use those data elements. This allows the Java class to focus on the operations needed instead of data types being used. This changes the focus from the language definition of possible data types to the functionality of the application being built. We use the term “**instance variables**” to refer to the data elements and the term “**methods**” to refer to the operations.

This program will need to have the Java Objects/Classes described above defined. The Java Class must have the **3 instance variables** of

- the 4-character guess pattern,
- the head count and
- the tail count.

The Java Class will also need multiple **methods** to access and update the instance variables. These methods are to include:

- void initialize (String) – which will store the given string a the 4-character guess pattern and set the head count and tail count to zero.
- char predictUserGuess ( ) – which will return the computer's next prediction based on the current values stored in the object's head count and tail count.
- void updateCount (char) – which will update the head count or the tail count based on the character value given as the parameter.
- String displayInfo ( ) – which will return a String showing the objects 4-character guess pattern and the current head count and tail count values.

All instance variables MUST be **private**. While the methods are most often **public**. The initialize( ) method described above could be implemented using a constructor; however, we may not get around to describing the special constructor methods before this project is due. Hence it is referred to as initialize( ).

The main method for this program should not be in the same class as the Java class described above. The lay out of the program with its minimum of 2 classes will be something like the following, which would be all stored in a single file called: Ptroy1Lab10.java

```
public class Ptroy1Lab10
{
  public static void main (String[] args)
  {
    guessTracker arr[];
    …
  }
} // end of Ptroy1Lab10 class

class guessTracker
{
  private String guessPattern;
  private int headCount;
  private int tailCount;

  public void initialize (String pattern)
  {
  }

  public char predictUserGuess ()
  {
  }

  public void updateCount (char g)
  {
  }

  public String displayInfo ()
  {
  }
}  // end of class guessTracker
```

Another java object/class is going to be highly encouraged that will keep track the last four guesses made by the user. This class would be used to store the current **state** of the program (the state of this program changes as the user makes his/her guesses). Using a java class to store and maintain state information is extremely helpful while programming. It allows all of the state data and operations needed to be performed on that state to be contained in a specific section of code.

Program Submission

You are to submit the programs for this lab via the Assignments Page in Blackboard.

To help the TA, name your file with your net-id and the assignment name, like:

- ptroy1Lab10.java