***The* University *of* Western Ontario**
CS1026a - Summer 2015
Assignment 1

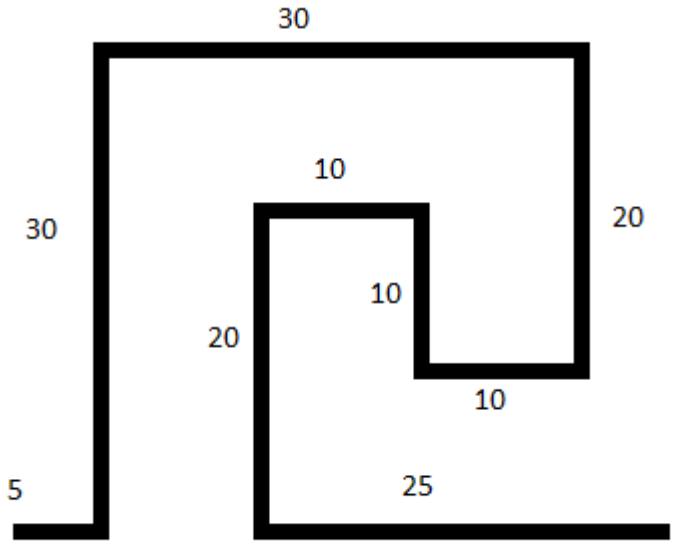
DUE: Thursday, May 28th, 11:55pm
Weight: 10%

Purpose:
To gain experience with

- Writing methods in Java
- Invoking methods
- Passing parameters to methods
- Using loops

Task:

In this assignment you are two write a complete program in java which uses the *Turtle* class to draw shapes on the screen. Your task is to draw a series of Greek 'keys' (see image below) with the goal of producing a large pattern. This assignment is broken up into 6 parts.



BEFORE YOU START:

Be sure that you have properly linked DrJava to the classes provided by the book (see DrJava instructions under resources on OWL). It might also be in your best interest to make a backup of the Turtle.java file in case you accidentally break it.

You will be creating your own class called *Assign1* and you will also be adding new methods to the already existing *Turtle* class. You will call the newly created methods in the *Turtle* class from your *Assign1* class. Below is the code I would like you to use to create your own class called *Assign1*:

```
public class Assign1
{
  public static void main(String[] args)
  {
    World w1 = new World();
    Turtle t1 = new Turtle(w1);

    t1.setColor(java.awt.Color.BLACK);

    t1.penUp();
    t1.moveTo(200,200);
    t1.penDown();

    //NEW STUFF HERE

  }
}
```

Within this main method we are:

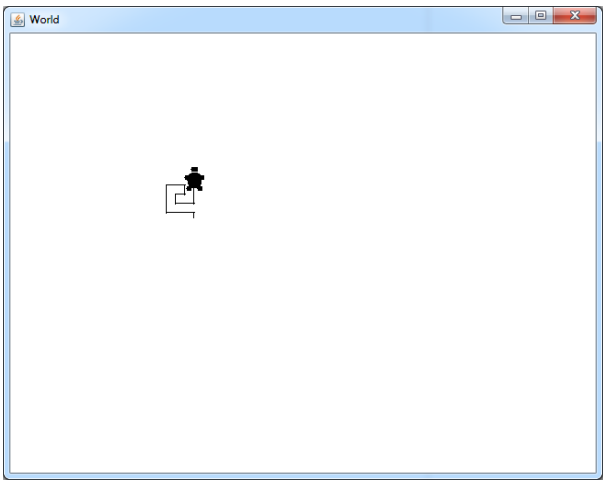
1. Creating a World object called w1
2. Creating a Turtle object called t1
3. Setting the colo***u***r of the turtle instance 't1' to BLACK
4. Lifting 't1's pen
5. Moving 't1' to location 200,200
6. Putting 't1's pen back down

PART 1:

You will write the following new method for the Turtle class in your *Turtle.java:*

```
public void drawKey()
```

The purpose of this method is to draw the key with the dimensions above. Note that the total width of the key is 40, the height is 30, and all turns are 90 degrees. The result should look like the image below. Note that sometimes the turtle's corners aren't as pretty as we would like (if yours has rough corners, don't worry).

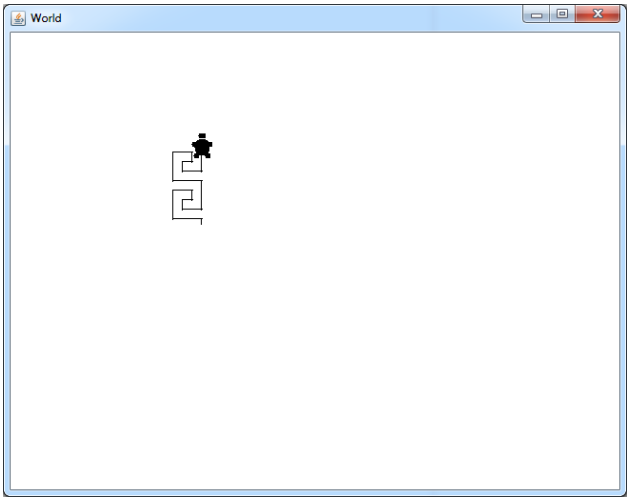


PART 2:

You will write the following new method for the Turtle class in your *Turtle.java:*

```
public void drawKeyRow(int numKeys)
```

The purpose of this method is to draw a row of keys. The parameter *numKeys* passed to the method will specify how many keys we want in the row of keys. In other words, if *numKey* is 2, then we want to draw two keys one after another. This method will call the `drawKey` method from Part 1. The image below is the result of the function being called with **2** being passed as the parameter.

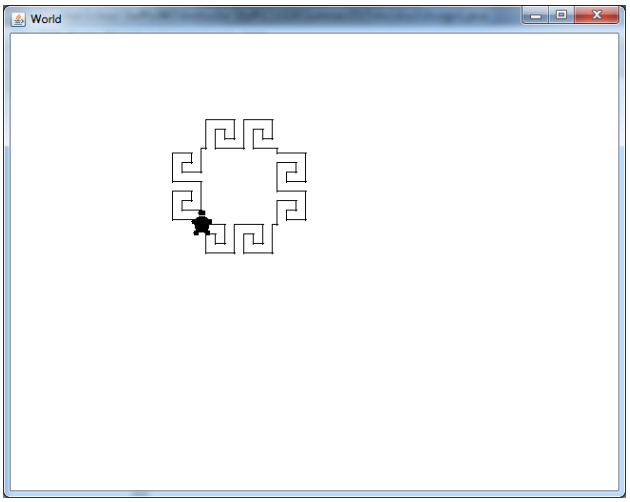


PART 3:

You will write the following new method for the Turtle class in your T*urtle.java:*

```
public void drawKeyFrame(int numKeys)
```

The purpose of this method is to draw a frame using the rows of keys. The parameter *numKeys* specifies how many keys we want in the row of keys. All frames have **4** edges. This method will invoke the method from part 2. The image below is the result of the function being called with 2 being passed as the parameter.

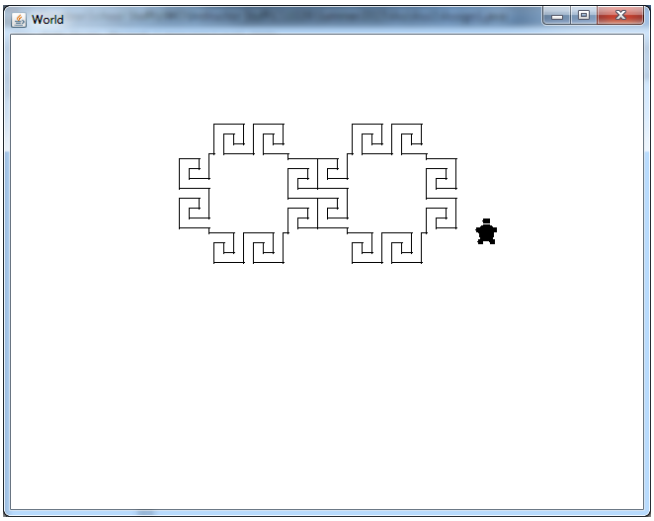


PART 4 (Here is where things start to get a little tricky):

You will write the following new method for the Turtle class in your T*urtle.java:*

```
public void drawKeyFrameRow(int numKeys, int numFrames)
```

The purpose of this method is to draw a row of frames. The parameter *numKeys* specifies the number of keys in each row of keys and the parameter *numFrames* specifies the number of frames to be in the row of frames. The image below is the result of the function being called with **2** and **2** being passed as parameters respectively. This method will require clever arithmetic (I hope you remember gr. 8 geometry). This method will invoke the method from part 3.

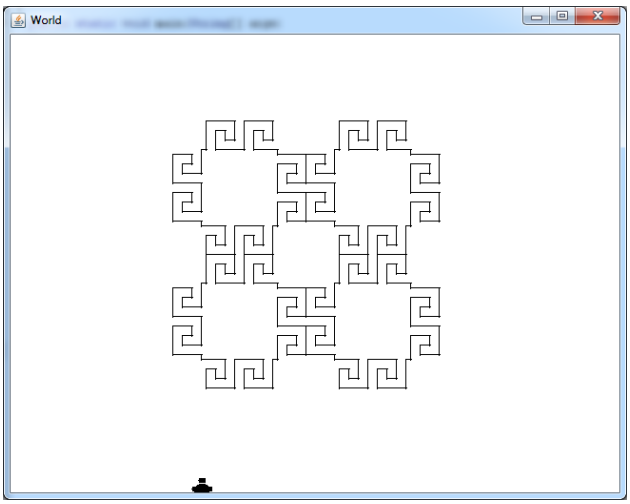


PART 5:

You will write the following new method for the Turtle class in your T*urtle.java:*

```
public void drawKeyFrameGrid(int numKeys, int numFramesRow, int numFramesCol)
```

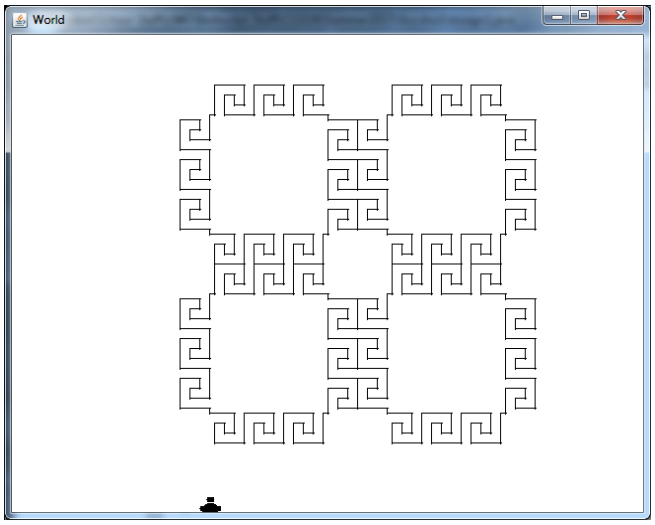
The purpose of this method is to draw multiple rows of frames. The parameter *numKeys* specifies the number of keys in each row of keys, the parameter *numFrames* specifies the number of frames to be in the row of frames, and the parameter *numFramesCol* specifies how many rows of frames we want (the number of columns in the grid). The image below is the result of the function being called with **2**, **2**, and **2** being passed as parameters respectively. Just like part 4, this part will require some arithmetic. This method will invoke the method from part 4.



Your `drawKeyFrameGrid` method must work with alternative parameters; although, it is acknowledged that the world may not be big enough by default to accommodate any parameter value. *You do not need to change the default world size for the purpose of this assignment.*

PART 6:

You are to invoke the `drawKeyFrameGrid` method from your main method within the *Assign1* class. Assign values 3, 2, and 2 respectively for the parameters. In other words, make *numKeys* = 3, *numFramesRow* = 2, and *numFramesCol* = 2. Your result should look something like the image below:



Now you are to copy the following code into your *Assign1* class below the main method (before the last '}' bracket) and complete it:

```
public static void drawPic(Turtle inTurtle, int x, int y, java.awt.Color inColour)
{
  inTurtle.penUp();
  inTurtle.moveTo(x,y);
  inTurtle.penDown();
  inTurtle.setColor(inColour);

  //ADD YOUR OWN CODE HERE
}
```

If you are having trouble knowing where to paste the above code, refer to the *Topic3Stuff.java* file under the "Resources/Lecture Topics/Lecture Code/Topic3" folder on the OWL website.

The first line of code within this class method lifts the pen up on the `inTurtle` object provided as a parameter. The second line moves the `inTurtle` to the `x` and `y` coordinates provided as parameters. The third line of code puts the `inTurtle`'s pen down, and the last line of code turns the turtle to the colour specified in the last parameter.

Add your own code below the first 4 lines to draw a simple picture. You achieve this by telling `inTurtle` to perform object methods (the methods within the *Turtle* class, ex. forward, turn, etc.). Your picture does not have to be complicated. Don't make the picture too large; the pictures will need to fit within each of the four frames.

You now must call this static class method from your main method 4 times; once for each frame. If you followed all the instructions up until this point correctly you can use the following line of code to draw the first picture; however, you may need to slightly alter the x and y positions (220 and 140 respectively) to make your picture fit within the frame. You may also need to alter the code within the `drawPic` method you just wrote in order to make the picture small enough to fit within the frames.

Copy the following line of code below your `drawKeyFrameGrid` method call in the main method of the *Assign1* class:

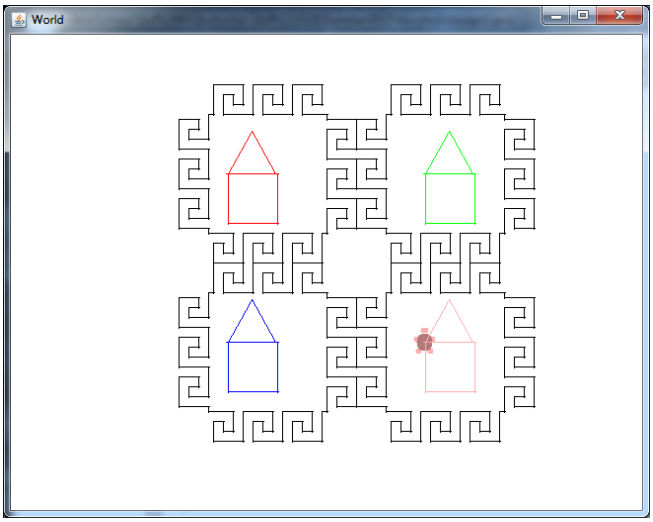
```
Assign1.drawPic(t1, 220, 140, java.awt.Color.RED);
```

Once you have one of the pictures working, all you will need to do is call it 3 more times. Each call will require different `x` and `y` values and ***MUST*** have a different colour.

```
Assign1.drawPic(t1, ???, ???, java.awt.Color.???);
Assign1.drawPic(t1, ???, ???, java.awt.Color.???);
Assign1.drawPic(t1, ???, ???, java.awt.Color.???);
```

All you will need to do is change the `x` and `y` coordinated and type a different colour in each of the method calls. Just change the `???`s to appropriate values.

Below is an example I did which drew a picture of a house in each of the four frames. I used RED, GREEN, BLUE, and PINK as my colours.



Non-functional Specifications:

1. Include brief comments in your code that explain what each method is doing.
2. Assignments are to be done individually and must be your own work. Software will be used to detect cheating.
3. Use Java conventions and good Java programming techniques, for example:
   i. Meaningful variable names
   ii. Conventions for naming variables and constants
   iii. Use of constants where appropriate
   iv. Readability: indentation, white space, consistency

Important! You must follow all the specifications above for your program

What You Will Be Marked On:

1. Functional specifications:
   - ✓ Are the required methods written according to specifications?
   - ✓ Do they work as specified? Pay attention to details!
   - ✓ Are they called as specified?
   - ✓ Are the objects drawn according to the specifications?
2. Non-functional specifications: as described above

Assignment submission: via the OWL, though the assignment submission in OWL.

What to Submit:

*Submit .java documents. Be VERY careful **not** to submit the .java~ version or the .class file. This can be tricky, as your system may not show the file extension in the file selection dialog.*

Double check your filenames on OWL before submission

1. Turtle.java (with your methods added!)
2. Assign1.java

How to Submit:

You are to submit the specified files through the online OWL submission under the "Assignment Details" tab.