

# CSc 110 Assignment 1: Introduction to Programming

## Learning Outcomes:

When you have completed this assignment, you should understand:

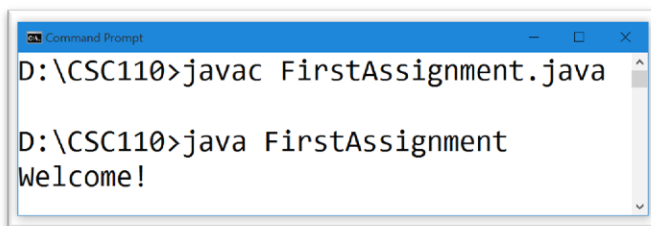
- How to design, compile, run and check a simple and complete Java program on your own.
- The effect of escape sequences on printed strings.
- How to write basic static methods.
- The flow of control (i.e. the effects of method calls and assignment statements).
- How to format and document a Java program.

## Part 1 – ASCII Art

In this part of the assignment, you will write the Java code that prints out a simple totem pole using American standard keyboard (ASCII) characters. You will create the code, in three separate static methods, `printPig`, `printFrog` and `printTotemPole`. You may use more methods and call them what you like, but these three must be spelled exactly as shown. The `main` method will direct the flow, by calling these three methods.

### Detailed steps

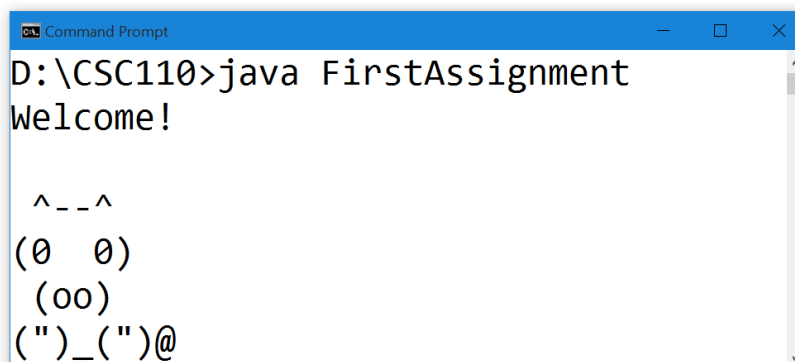
1. Create the public class called `FirstAssignment.java`.
  - a. Save and compile this file, making sure there are no errors before proceeding.
2. Add a `main` method.
  - Method header: `public static void main(String[] args) {`
  - Inside this method, write a statement that prints out “Welcome!” to the console
  - Save, compile, and run the program. Repeat until error-free and output is similar to the following image:



```
Command Prompt
D:\CSC110>javac FirstAssignment.java

D:\CSC110>java FirstAssignment
Welcome!
```

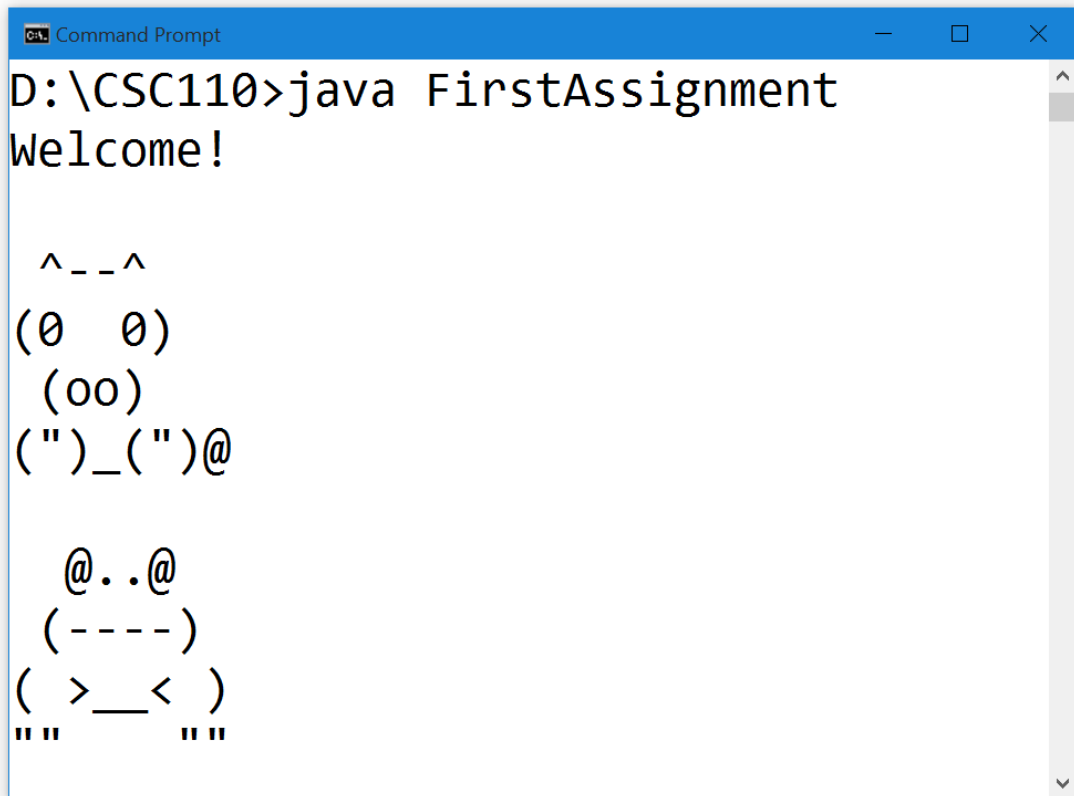
3. Write and test the `printPig` method.
  - Method header: `public static void printPig() {`
  - Call this method from the `main` method.
  - Save, compile and run the program. Repeat until error-free and output is similar to the following image:



```
Command Prompt
D:\CSC110>java FirstAssignment
Welcome!

  ^ _ ^
(0  0)
 (oo)
(")_(")@
```

4. Write and test the `printFrog` method.
- The header is similar to the `printPig` header.
  - Call this method from the `main` method.
  - Save, compile, and run the program. Repeat until error-free and output is similar the following image:



```
D:\CSC110>java FirstAssignment
Welcome!

      ^ _ _ ^
    (0  0)
     (oo)
    (" )_ (" )@

      @ . . @
    ( - - - - )
   (  > _ <  )
  " "      " "
```

5. Write and test the `printTotemPole` method.
- In `printTotemPole`, call `printFrog` and `printPig` to stack the animals as shown in the image on the next page.
    - Note that there is a *spacer line* between each of the animals.
    - Note also that you will have to modify `printPig` so it sits evenly on the totem pole.
  - Replace the calls to `printPig` and `printFrog` in the `main` method with a call to `printTotemPole`.
  - Save, compile and run the program. Repeat until error-free and output is similar to the image on the next page:

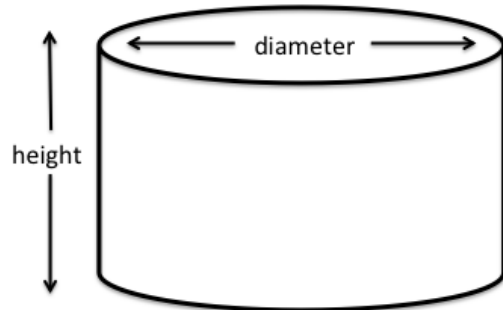
```
Command Prompt
D:\CSC110>java FirstAssignment
Welcome!

/~~~~~\
 @..@
 (----)
 (>__<)
""""""
/~~~~~\
 ^__^
 ( 0 0)
 (oo)
 (")_(")@
/~~~~~\
 @..@
 (----)
 (>__<)
""""""
/~~~~~\
 ^__^
 ( 0 0)
 (oo)
 (")_(")@
/~~~~~\
/~~~~~\
```

NOTE: These images are done on a Windows computer using the Consolas font, different fonts may result in slightly different outputs, you will not be penalized for this, don't worry!

## Part II – Math Calculations

In the same `FirstAssignment.java` file, you are to write code to calculate the surface area of a closed cylinder that has a given height and diameter. The formula for the surface area is twice the area of a circle (the top and bottom), added to the area of the rectangle when you cut and flatten the *wall* of the cylinder. See the diagram below:



Rather than write out the whole formula in one step, you will break it down into smaller steps, storing each result into a variable that you create and name appropriately. Each subsequent step will use only the variables and NOT the literal values. You will be successful if the calculations are correct AND there is only one '4', one '5' and maybe a '2' or two in the code.

### Detailed steps

1. Create another method inside `FirstAssignment`, called `calcSurfaceArea`.
  - Declare two integer variables: `height` and `diameter`. Assign the literal values 5 to `height` and 4 to `diameter`. (That will be your quota for '4' and '5' literals.)
  - Create a new variable called `radius` that is one half the `diameter`. (One of the '2's.)
2. Write a single statement to calculate the circumference of the cylinder, and assign the result to a new variable, appropriately named. Because  $\pi$  (pi) is required for any calculations involving circles, you will need to use a variable for pi. Luckily, Java has one in its library, called `Math.PI`, which provides an accurate value.
3. Write a single statement to calculate the area of the rectangle that forms the wall of the cylinder with the given `height` and `diameter`. Assign the result to a new variable, appropriately named.
4. Write a single statement to calculate the total area of a circle with the given `diameter`, and store the result in a new variable, appropriately named.
  - Computers are faster at adding than multiplying, and are faster at multiplying than calculating to a power, so most programmers will write `radius*radius`, instead of `Math.pow(radius,2)`. You may use either.
5. Write a final statement that stores the total surface area into a variable. It will sum the two circle areas and the wall area.
6. Print the resulting value of the surface area to the console.
7. Call `calcSurfaceArea` from the `main` method.
8. Write, compile and run your program until your output looks similar to image on the following page...

```
Command Prompt
D:\CSC110>java FirstAssignment
Welcome!

/~~~~~\
 @..@
 (----)
 ( >__< )
 ""__""

/~~~~~\
 ^__^
 ( 0 0)
 (oo)
 (")_("@)
/~~~~~\
 @..@
 (----)
 ( >__< )
 ""__""

/~~~~~\
 ^__^
 ( 0 0)
 (oo)
 (")_("@)
/~~~~~\
/~~~~~\

Total surface area is: 87.96459430051421
```

### How to hand it in:

Submit the single file with your finished *source* code (`FirstAssignment.java` --NOT `FirstAssignment.class`) to the CSC110 [conneX](#) site as an attachment in the Assignment 1 page.

### Grading:

Marks will be allocated for the following...

- Your code must compile and run.
- Your code should produce the output requested.
- Your code must be indented and documented appropriately. Please follow the guidelines in [Style\\_Guidelines.pdf](#), available in the Resources folder on [conneX](#).