

# CSC110 Assignment 1:

## Introduction to Programming

### Objectives

Upon completion of this assignment, you need to be able to:

- Design, compile, run and check a simple and complete Java program on your own.
- Use the escape character where necessary to print out some characters of a string.
- Do a little basic computation, using *variables* to store values.
- Write a basic Java method.
- Experience proper control flow by using method calls and assignment statements.
- Format and document Java source code.

### Introduction

ASCII art was a simple technique for early programmers to make little drawings, before the days of computer graphics. We will use this technique to get some output to the console after creating the source code, compiling the code and then running it.

In the second part of this assignment, we will do some very basic *computing*, a task computers were initially designed for.

## Quick Start

- (1) Download this pdf file and store it on your computer.
- (2) Create a directory / folder on your computer specific to this assignment. for example, CSC110/assn1 is a good name.
- (3) Save a file called `FirstProgram.java` in your directory. Fill the source code with the class name and the following methods:
  - `public static void main(String[] args)`
  - `public static void printFrog()`
  - `public static void printOwl()`
  - `public static void approxPi()`Make the methods empty by adding a `{` and a `}` with only blank space between them. Blank spaces include tabs, spaces and newlines (also known as carriage returns).
- (4) Compile and run the program frequently, preferably after each of the detailed instructions below.

## Detailed Instructions

### Part I : Ascii Art

Inside the method called `printFrog`, write the statements that will produce the following output using `System.out.println` statements:

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

Inside the method called `printOwl`, write the statements that will produce the following output using `System.out.println` statements:

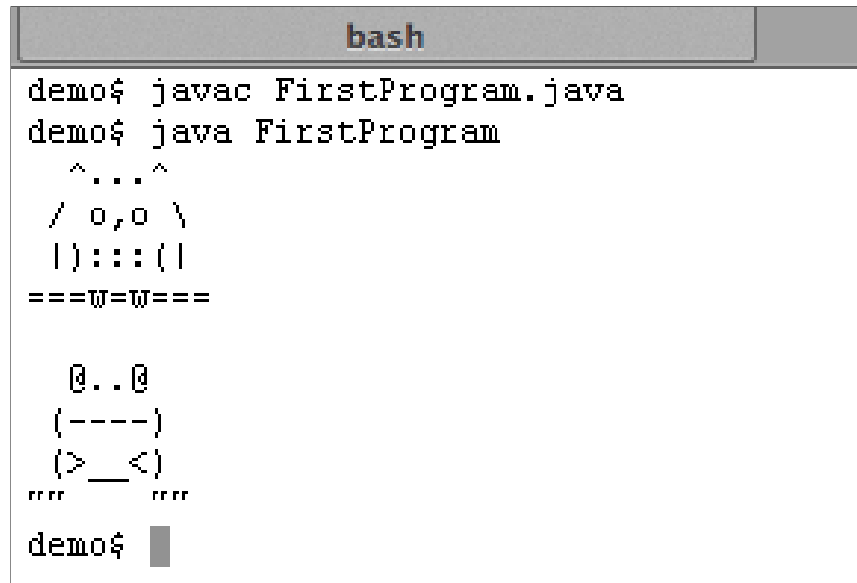
```
^...^
/ o,o \
|)::(|
===w=w===
```

Test both `printFrog` and `printOwl` by printing the following statements inside the braces of the `main` method of the source code:

```
printOwl();
printFrog();
```

Add a line or two for space between the two drawings.

The following screenshot shows an example of the compilation of the source code, the running of the program and the output on the console:



```
demo$ javac FirstProgram.java
demo$ java FirstProgram
  ^...^
 / o,o \
 |):::(|
===W=W===

  @..@
 (----)
 (>__<)
'''  '''
demo$
```

## Part II : Approximating $\pi$

The number Pi ( $\pi$ ) is the ratio of any circle's circumference to its diameter. In many programming libraries, this number is approximated with the following calculation:

$$\pi = 4 \times (1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - 1/15 + \dots)$$

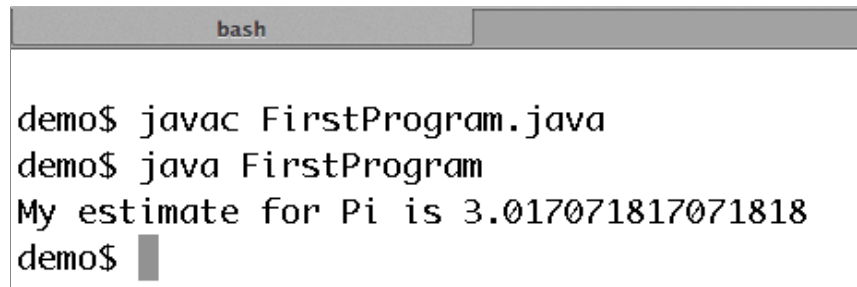
the dots meaning that the series goes on infinitely.

Inside the `approxPi` method, write the Java statements that will approximate the value of  $\pi$  using the first 8 terms of the sequence.

Although this can be done by simply typing in the numbers  $1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - 1/15$  directly into a single statement and then multiplying by 4, you must not submit this version. Instead, create the source code using the following steps:

- (1) Create a variable called `nextTerm` and initialize it to the value 1.
- (2) Create a variable called `denom`, short for *denominator* and initialize it to the value 1.
- (3) Create a variable called `series` and initialize it to the value 0.
- (4) Do the following eight times:
  - (a) Add (`nextTerm/denom`) to `series`
  - (b) Add 2 to `denom`.
  - (c) Multiply `nextTerm` by  $-1$ .
- (5) Multiply `series` by 4 and print the result.

Test this method by commenting out the previous statements in the `main` method and adding a call to `approxPi`. Now, when you compile and run, you will see the following:



```
bash
demo$ javac FirstProgram.java
demo$ java FirstProgram
My estimate for Pi is 3.017071817071818
demo$
```

## Submission

Submit the following completed file to the Assignment folder on `conneX`. You will be shown how to do this on the first week of labs.

- `FirstProgram.java`

Please make sure you have submitted the required file(s) and `conneX` has sent you a confirmation email. Do not send `[.class]` (the byte code) files. Also, make sure you *submit* your assignment, not just save a draft. Draft copies are not made available to the instructors, so they are not collected with the other submissions. We *can* find your draft submission, but only if we know that it's there.

## **A note about academic integrity**

It is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement their own solution.

## **Grading**

Marks are allocated for . . .

- No errors during compilation of the source code.
- Output produced exactly as requested in this document.
- Methods are written as specified.
- Methods are all tested in `main`; they may be commented out but not removed.
- Style of the source code meets the requirements outlined in the [codingConventions](#) document available in the Resources folder of `conneX`.