Project 1: Console-based Plotting

1 Task

Write a program to do console-based plotting of curves/lines in quadrants I and II, but *sideways*. Write four functions, named (exactly): plotXSquaredPlus2, plotAbsXTimes3, plotNegXSquaredPlus25, plotSinWave, and main. The main method should call the others and feed them hardcoded data (since we have not yet covered user input). Each function should accept two integer parameters indicating the minimum and maximum X value to be plotted. The functions should have no return values. Create class-level constants for things we might wish to change later, e.g., a *char* called PLOT_CHAR that holds an “x” (per samples below), another char called FILL_CHAR that holds an unobtrusive character like ASCII 183, generated with a cast, e.g., (char)183.

2 Output

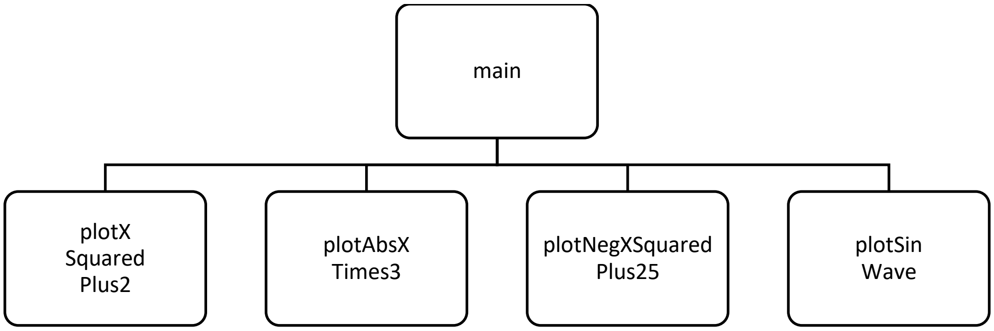
When functions are called with indicated X values, output should be as shown on the next page. In the functions, construct subtitles using passed-in data; do not hardcode ranges (we might change them later). The area under the plot is estimated by summing the y values; this is a rudimentary Riemann sum (more later).

3 Code Implementation

Create a class called **SidePlot** with all functions within that class. Follow the Course Style Guide (always).

3.1 Call Hierarchy

Create the following public static functions. Use this exact call hierarchy (i.e., second-tier functions must *not* call each other; that would be *chaining,* and we never want that):



3.2 What You **May** Use

- Constants (at class level), using the requested naming convention (see the Course Style Guide)
- Variables (but not class-level ones), using requested naming convention (camelCasing)
- Assignment and calculations
- Definite loops (for); note that each graph should require no more than *two* for statements
- Console output
- Additional helper functions to reduce redundancy

3.3 What You **May Not** Use

- Indefinite loops (while); they are not helpful or indicated here, anyway
- Selection control structures (if) unless you are doing the extra credit
- Arrays, lists, or other data structures, objects, libraries, or methods we have not covered

4 Output

```
Sideways Plot
y = x*x+2 where -5<=x<=5
···························x
··················x
···········x
······x
···x
··x
···x
······x
···········x
··················x
···························x
The area under the plot is 132


Sideways Plot
y = |x|*3, where -6<=x<=6
··················x
···············x
············x
·········x
······x
···x
x
···x
······x
·········x
············x
···············x
··················x
The area under the plot is 126


Sideways Plot
y = -(x*x)+25 where -5<=x<=5
x
·········x
················x
·····················x
························x
·························x
························x
·····················x
················x
·········x
x
The area under the plot is 165


Sideways Plot
y = 20sin(.5x)+20 where -11<=x<=11
··································x
·······································x
·······································x
···································x
···························x
·················x
········x
·x
x
···x
··········x
····················x
·····························x
····································x
·······································x
······································x
·······························x
······················x
············x
····x
x
x
·····x
The area under the plot is 449
```

5 Extra Credit A

Instead of the output shown above, produce this output which includes axis markers and a glimpse into quadrants III and IV. You are allowed selection control structures for this work.

```
Sideways Plot
y = x*x+2 where -5<=x<=5
  |··························x
  |·················x
  |··········x
  |·····x
  |··x
--+-x--------------------------
  |··x
  |·····x
  |··········x
  |·················x
  |··························x
The area under the plot is 132


Sideways Plot
y = |x|*3 where -6<=x<=6
  |·················x
  |··············x
  |···········x
  |········x
  |·····x
  |··x
--x-------------------
  |··x
  |·····x
  |········x
  |···········x
  |··············x
  |·················x
The area under the plot is 126


Sideways Plot
y = -(x*x)+25 where -5<=x<=5
  x
  |········x
  |···············x
  |····················x
  |·······················x
--+------------------------x-
  |·······················x
  |····················x
  |···············x
  |········x
  x

The area under the plot is 165


Sideways Plot
y = 20sin(.5x)+20 where -11<=x<=11
  |·································x
  |······································x
  |······································x
  |··································x
  |··························x
  |················x
  |·······x
  |x
  x
  |··x
  |·········x
--+-------------------x--------------------
  |····························x
  |···································x
  |······································x
  |·····································x
  |······························x
  |·····················x
  |···········x
  |···x
  x
  x
  |····x
The area under the plot is 449
```

6 Extra Credit B

Write a separate class called Limits. In it, write a main function that uses definite loops to check the following statement:

> *For a loan of $100,000 at an annual interest rate of 2.5%, the limit of the total amount paid as the number of payments approaches infinity is $101,255.20*

Output should start like this:

```
# Pmts Total Paid
1      102500.0
10     101380.15
100    ...
```

To accomplish this task, you will need several building blocks:

- The formula for calculating a loan payment amount is shown below. We would normally round to the nearest penny—you cannot make payments in fractional cents—but leave it at full precision instead

$$P = \frac{r(PV)}{1 - (1 + r)^{-n}}$$

$P = Payment$
$PV = Present\ Value$
$r = rate\ per\ period$
$n = number\ of\ periods$

- The formula for calculating the total amount paid: TP = P(n)
  - Round this to the nearest penny before display
- A Java method that rounds to the nearest integer: Math.round(*number*)
  - Note that there is no second parameter; you will need to get creative
- A Java method that raises a number to a power: Math.pow(*number*, *exponent*)
- Incrementing your loop variable logarithmically; we want to see the number of payments and the total of payments for 1 payment a year, 10 payments a year, etc., through 1,000,000,000 payments a year (10^9).

7 Submitting Your Work

Submit the **.java file(s)** from your project; there is no need to send other files or zip up the folder.

8 Hints

There is a good bit of duplicated code between the plotting functions; do not worry about it in this assignment. If you are clever, you might create a helper function or two that might reduce duplication, e.g., to help build the display header for each graph.

You do not need to guard against bad data coming in via parameters (in later work, you will). But you should test your program with values that are in the specified ranges that should work.

9 Grading Matrix

Area	**Points**
Class	5%
Functions:	
Definition & calling	20%
Headers built & shown	10%
Looping	20%
Parameter use	15%
Areas calculated/shown	10%
Main	10%
Documentation/style	10%
Extra credit A	5%
Extra credit B	2%
Total	**107%**