UNIVERSITY OF ALBERTA

AUCSC 112 LAB

Assignment #1

(Due before midnight on day before the next lab)

Goals:

- To program in Java – use sequencing, simple methods, and simple loops (use `while` and `for-each` only).
- To increase in computational thinking ability, and be able to explain why code works.
- To name descriptively, to maintain proper code indentation, and to begin to subdivide program code.
- To distinguish data types, use them (declare variables), and convert between them.

Reference:

Heise, chapters 0 and 1.

Instructions:

Write a variety of Java methods and code, as outlined below (Part 1, Part 2, and Part 3), in three separate files (and separate projects). You may always create more than the requested number of methods; you are even encouraged to do so.

This is an INDIVIDUAL assignment - the code you write should be entirely your own. Do not use code from anyone else (including from the internet). Do not submit any parts of this assignment to any internet site, including homework help sites. When you are helping someone else, do not show your own code. You may help others by finding bugs or explaining a concept, but do not share code.

Do not import any libraries. Do not access Java's String library methods (i.e. no dot messages to Strings), except the concatenation operator (+) is allowed.

Ensure that you name all files and methods EXACTLY as specified, remembering that Java is case-sensitive.

Part 1:

Filename: **`Main.java`**
This file must contain the following components:

1) A file header with information about the code, including file summary, author, class, ID number, and date. The file header should be the first thing in the file.

2) A method: **`main(String[]) → void`**
   This method prints "☺ Hello Augustana Programmer ☺" to the IDE's output window. A smiley can be made by copying the symbol or with `\u263A`.

3) A method: **`singHappyBirthday(String) → void`**
   This method takes someone's name and prints out the birthday song, with that person's

name filled in.

For example `singHappyBirthday("Mickey");` must produce:

```
🎂 Happy Birthday to you!
🎂 Happy Birthday to you!
🎂 Happy Birthday, dear Mickey.
🎂 Happy Birthday to you! 🎶
```

Note that the formatting is important. Do not have extra or fewer spaces, and include punctuation and symbols (the cake is `\uD83C\uDF82` and the musical notes are `\uD83C\uDFB6`).

If the name is `null` or the empty string, simply print "`Nothing to sing.`" and no other letters or symbols.

Be certain to include a Javadoc method header (`/**…*/`).

4) Extend `main` so that it prints the Happy Birthday song to each element in this array:

```
String[] classList = {"Mickey Mouse", "Minnie Mouse",
                      "Donald Duck", "Daisy Duck",
                      "Pluto", "Santa Claus",
                      "Wile E Coyote", null, "Road Runner",
                      "Bugs Bunny", ""};
```

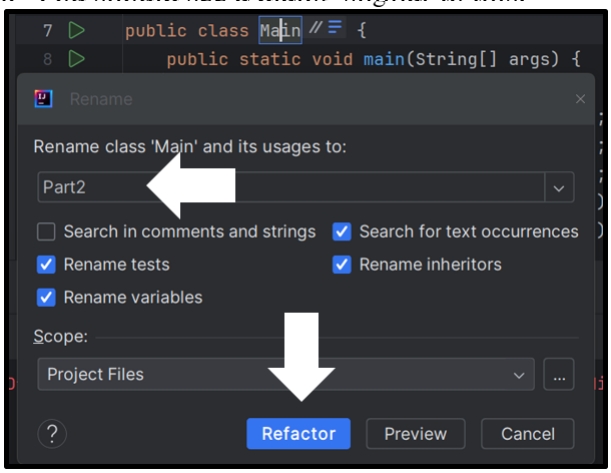
Do this by using Java's enhanced for loop, the for-each loop with a colon in the middle, which matches Python's most common loop. Do not use Java's iterated for loop.

Make sure you have a blank line between the messages to each person. This blank line should come from `main`, not from your `singHappyBirthday` method.

Part 2:

Filename: **`Part2.java`**

Note: to change the filename, click on the class name "Main", then from the top menu tabs select "Refactor" → "Refactor This…" → "Rename" → "Refactor" → "Refactor This…" → "Rename". (You need to do the sequence twice to get the actual dialog box to change both the filename and class name at the same time.) Put in the new name for the class (and file happens automatically) and click the "Refactor" button. You should see a dialog similar to this:



This file must contain the following components:

1) A file header with information about the code, including file summary, author, class, ID number, and date.

2) A method: **`main(String[]) → void`**
   This method will contain the testing that you write for the following two (or more) methods.

3) A method: **`printNumStrings(int, String) → void`**
   This method takes a number (as a primitive integer) and a String, and repeatedly prints that string the given number of times. Do not move to the next line. Make sure that your method handles sizes 0 or negative appropriately. For any loop that you create, use only a `while` loop.

   For example: `printNumStrings(4, "&");`
   `&&&&` and does not move to new line nor end with a space.

4) A method: **`printTriangle(int, char) → void`**
   This method prints a hollow triangle, with tip down, out of the character specified. The triangle must be of the size specified by the first parameter. This size gives both the number of characters on the first line and the number of lines (i.e. it is an equilateral triangle). Make sure that your method handles sizes 0 or negative appropriately. For any loop that you create, use only a `while` loop. You should make use of your `printNumStrings` method as much as possible. Note that characters may be converted to Strings with concatenation to the empty String: `'&' + ""` → `"&"`. If the size is less than 2 or bigger than 50 print an appropriate message.

   For example: `printTriangle(2, '&');`
   ```
   & &
    &
   ```

   For example: `printTriangle(0, '=');`
   ```
   Too small to print.
   ```

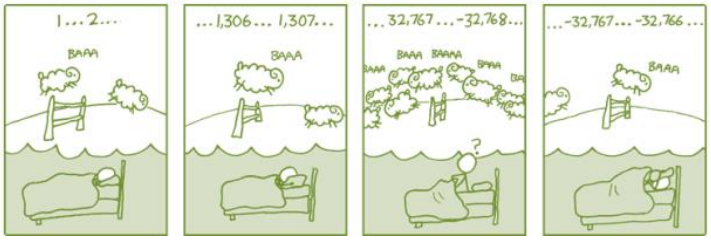
   For example: `printTriangle(51, '+');`
   ```
   Too big to print.
   ```

   For example: `printTriangle(5, '*');`
   ```
   * * * * *
    *     *
     *   *
      * *
       *
   ```

5) Other methods you might want to create, to make `printTriangle` more modular.

Part 3:

Filename: **`Part3.java`**
This file must contain the following components:



1) A file header and method headers for any methods that you make.

2) A method: **`main(String[] args)` → `void`**
This method contains the testing that you write for the following method.

3) A method: **`reverseInt(int)` → `int`**
This method returns the integer, with its digits reversed. Keep the sign the same, and do not convert the integer into a String at any point in your process. If the reverse of the number is too big or too small to fit into an integer, return Java's maximum integer (`Integer.MAX_VALUE = 2147483647`) if the initial integer was positive and Java's minimum integer (`Integer.MIN_VALUE = -2147483648`) if the initial integer was negative. (Hint: the modulus operator can be used to extract a particular digit, for example, 5723 % 10 will give the 3. You will need to use division and modulus in a repetitive pattern.)

   Here are some examples:

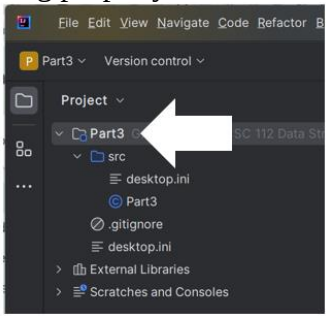
```
reverseInt(-5723) → -3275
reverseInt(123456) → 654321

reverseInt(120) → 21
reverseInt(0) → 0
reverseInt(10) → 1

reverseInt(-2147483641) → -1463847412
reverseInt(-1463847413) → -2147483648 (minimum int)
```

   Do not use the Math library, do not convert the integer to a string, and do not use any String methods except possibly concatenation. Notice also that this method needs to return a value (and not to print anything from the method – any testing that you do would print the returned result inside `main`).

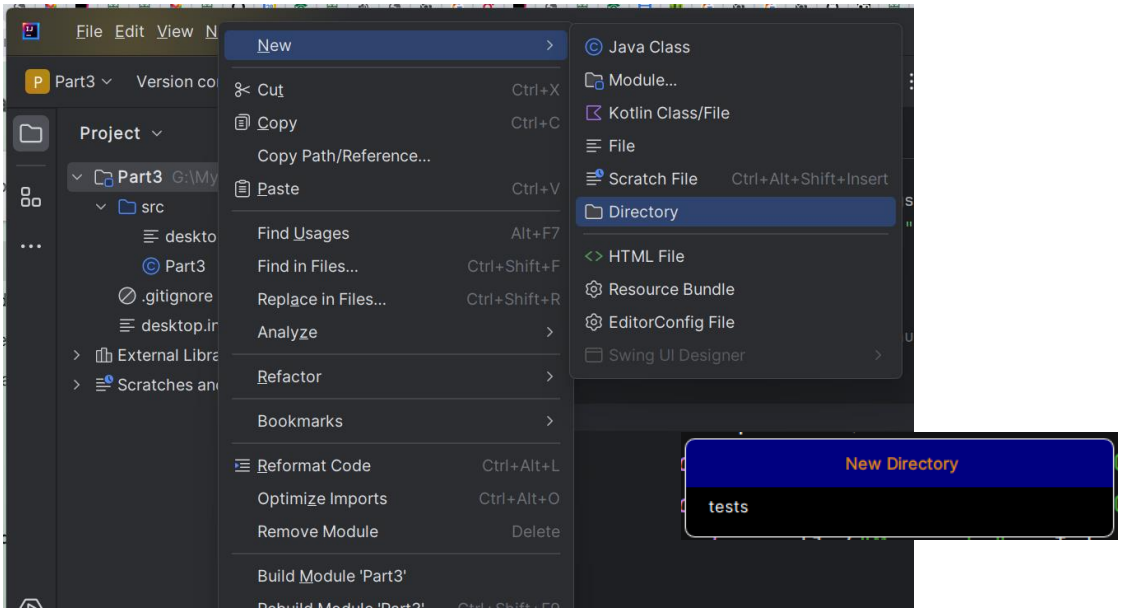
Testing `reverseInt`

Once you have made the `reverseInt` method, test it in the IntelliJ IDE, by creating a testing file "Part3Test.java", and copying in the contents of the same file from eClass. Here are instructions for how to set up and perform the testing properly in IntelliJ:

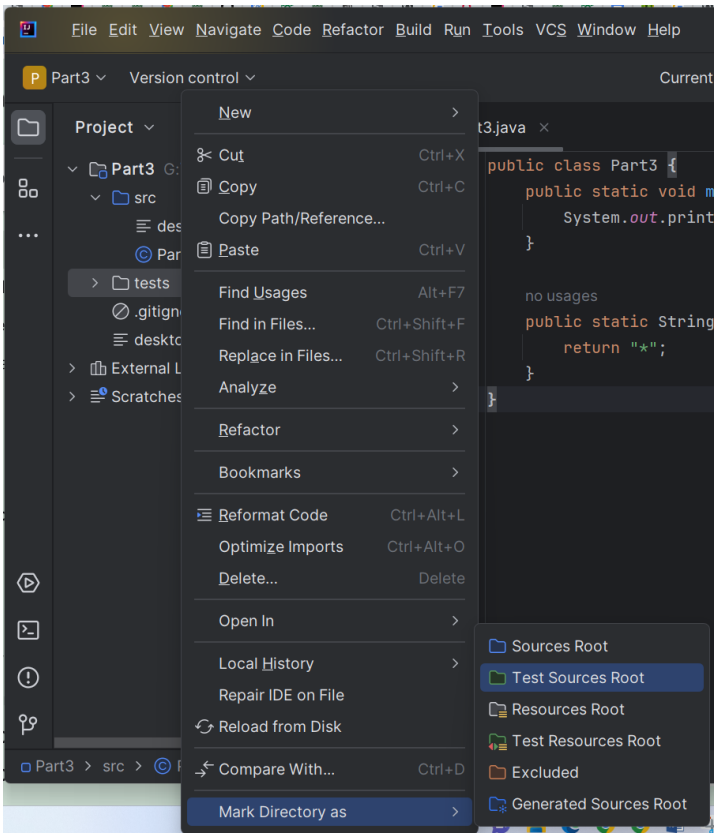
1) Make a directory for the testing:
   a) Right-click on your project at the left side:



b) “New” → “Directory” → fill in the directory name as “tests” and press “Enter”

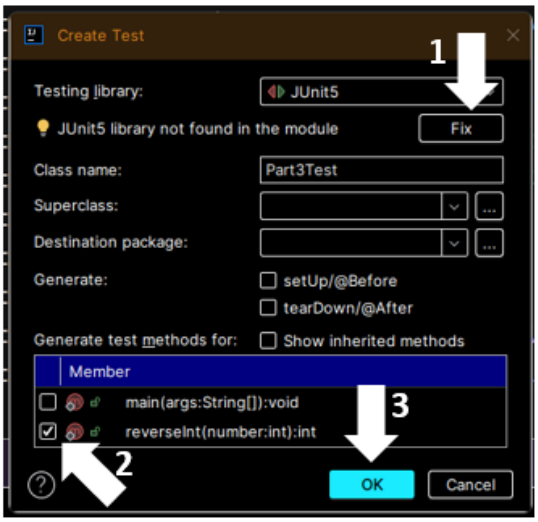


c) Right-click the tests directory → “Mark Directory as” → “Test Sources Root”



2) Make the testing file:
   a) Click on the class declaration line, “public class Part3”.
   b) Press Alt-Enter or Right-click and select “Show Context Actions”.
   c) Click “Create test”.
   d) Click the “Fix” button if the library is not found.

e) Select reverseInt, and click "OK".

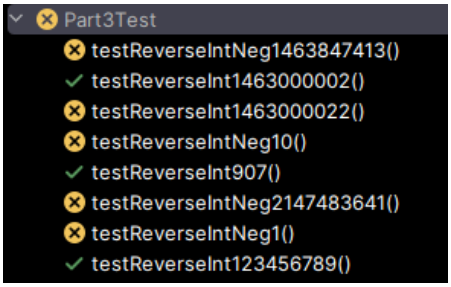


3) Copy the contents from eClass into this new file.

4) Run the tests by simply running the testing file. If all of the tests pass, then you will see something similar to this:

Otherwise, you might see something like this:

If any tests have failed, then you should modify your code to attempt to correct your solution. The bottom, leftmost window shows which tests have failed (with a yellow "X") and which have passed (with a green checkmark). For example:



You can run the tests as many times as you want. Note that you should *not* write code that only passes the given set of tests, but that rather works for all possible test cases, even those not included in this particular testing file. In other words, do not simply add a bunch of if-else statements giving the correct answer for the particular test cases in the testing file. This would be incorrect coding practice.

How to Hand In Your Work:

Submit six files on eClass.

1) Three `.java` files
   a) `Main.java` (from Part 1)

b) `Part2.java`

c) `Part3.java`

Remember that they must be named exactly with matching case and proper extension. The name of your class (inside the file) must match too.

2) A `.pdf` of each Java file, made through IntelliJ, in color, 8 pt font minimum, portrait orientation, narrow margins, and monospaced font.

a) `Main.pdf`

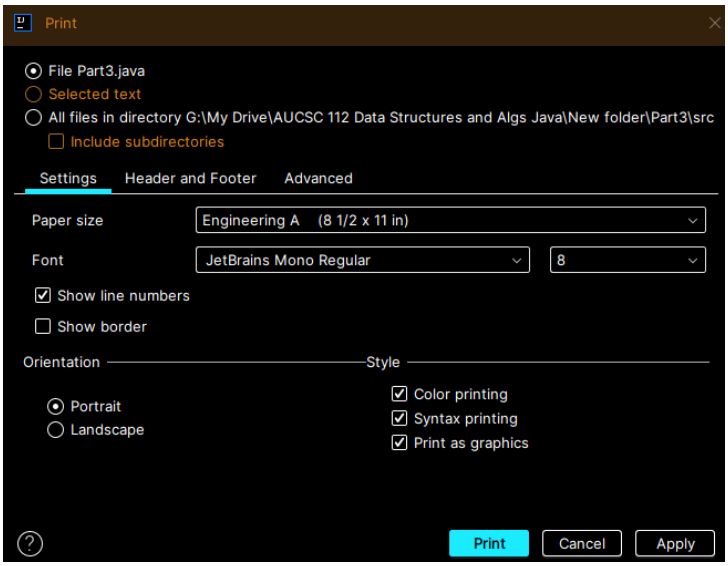
b) `Part2.pdf`

c) `Part3.pdf`

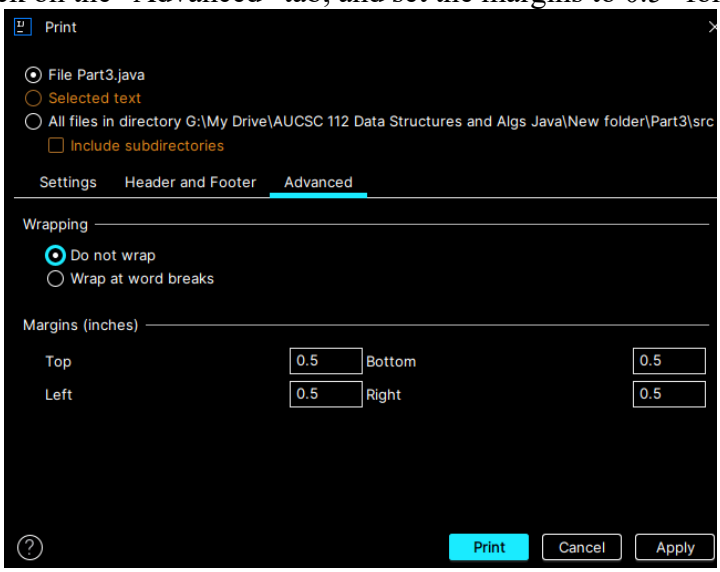
Here are the instructions for how to make a pdf using a Windows machine.

1) File → Print…
2) Now select the options for "Settings":
   a) Pick the Paper size to be 8.5" x 11".
   b) Set the font to "JetBrains Mono Regular" and size to "8".
   c) Unclick the "Print Border" box, put leave the line numbers set.
   d) Click "Color printing".

   The dialog box should look like this:

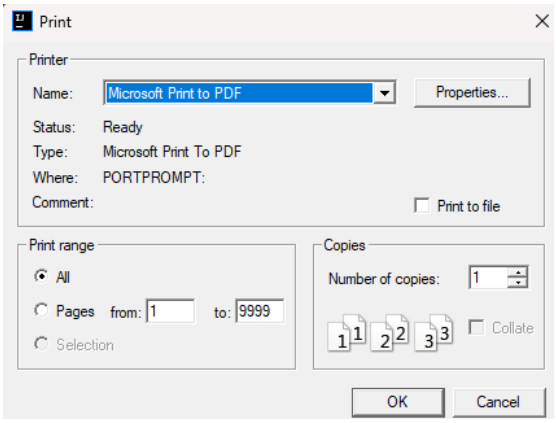


3) Now click on the "Advanced" tab, and set the margins to 0.5" for all.



4) Click on “Print”.

5) Select “Microsoft Print to PDF” and then “OK”.



6) Fill in the name of the file (match the Java name). Click “Save”.
7) Verify that your file is readable, and that there are no lines that are cut-off or wrapping in a way that messes up the indentation. [Note: if you have lines that are too long, then perhaps you forgot to set the line length in the editor settings.]

Practice Questions – Assignment 1 Concepts:

Do not hand in your solution to these. These questions are provided so that you can check whether you have learned the concepts we are expecting you to learn. No solutions are provided, and we encourage you to discuss these questions with other students. These are excellent questions to help you prepare for exams.

1) How do you start writing a Java program?
2) How is a comment put into a Java program? State two ways.
3) Where does program execution start in Java?
4) Why does the main method have a `void` return type?
5) What is `String[] args`?
6) What is the shortcut for running a Java program in your IDE?
7) What are three things you can say about `System.out.println`?
8) What is a method (= function = subprogram)? List all the methods you wrote.
9) What are the parameters to a method?
10) What is the formal parameter in your `singHappyBirthday` method? What is the actual parameter?
11) What is the difference between writing a method and calling a method?
12) How is the return type of a method specified?
13) What is the difference between printing from a method and returning a value from a method?
14) `classList` is an array. What does it mean to be an array? Draw a memory diagram for `classList`.
15) What is the data type of each element in `classList`?
16) What is a way to create repetition in Java code?

17) How do you make a Python-like for loop (called a for-each loop) in Java?
18) What is the difference between a for-each loop and a while loop?
19) What is `'\n'`? Where can you put `\n` to be meaningful in a program?
20) What does it mean to declare a variable?
21) What is the difference between a local variable and a parameter?
22) How is division in Java different from division in Python?
23) What is a syntax error? Give an example of a syntax error you made.
24) What is a run-time error? Give an example of a run-time error you experienced.
25) What is a logic error? Give an example of a logic error you made.
26) How do methods help the programming process?
27) Where is the source file of your programs located? Where is the byte code file located?
28) What is the difference between a `null` string and the empty string?
29) Suppose that an integer with magnitude greater than 10 is stored in the variable `x`. Write the Java code to get the 2nd last digit of this number.
30) Explain the comic at the top of Part 3.

Images Credits (all retrieved 23 January 2025):
Java: https://www.java.com/en/
Balloons: https://www.clipartbest.com
Triangle with sunglasses: https://www.charatoon.com/
Sheep counting: https://www.approxion.com/grokking-integer-overflow/
Question mark: https://www.freepik.com/free-photos-vectors/question-mark-cartoon