Com S 227
Spring 2022
Assignment 1
100 points

Due Date: Friday, February 11, 11:59 pm (midnight)

5% bonus for submitting 1 day early (by 11:59 pm Feb 10)

This assignment is to be done on your own. See the Academic Integrity policy in the syllabus, for details.
You will not be able to submit your work unless you have completed the *Academic Dishonesty policy questionnaire* on the Assignments page on Canvas. Please do this right away.

If you need help, see your instructor or one of the TAs. Lots of help is also available through the Piazza discussions.

Please start the assignment as soon as possible and get your questions answered right away. It is physically impossible for the staff to provide individual help to everyone the afternoon that the assignment is due!

This is a "regular" assignment so we are going to read your code. Your score will be based partly (about a third) on the specchecker's functional tests and partly on the grader's assessment of the quality of your code. See the "More about grading" section.

Contents

Tips from the experts: How to waste a lot of time on this assignment

1. Start the assignment the night it's due. That way, if you have questions, the TAs will be too busy to help you and you can spend the time tearing your hair out over some trivial detail.
2. Don't bother reading the rest of this document, or even the specification, especially not the "Getting started" section. Documentation is for losers. Try to write lots of code before you figure out what it's supposed to do.
3. Don't test your code. It's such fun to remain in suspense until it's graded!

Overview

The purpose of this assignment is to give you some practice with the process of implementing a class from a specification and testing whether your implementation conforms to the specification. You'll also get some practice with modular arithmetic.

For this assignment you will implement one class, called **Printer**, that models pages being printed from a simple printer. The printer has a paper tray that can hold up to some maximum number of sheets of paper. The tray can be removed (in which case there are no sheets of paper available for printing) and replaced (making the sheets available for printing). Paper can be removed from and added to the tray.

The printer receives a print job that is associated with a document of *n* pages, where the number *n* is specified by calling the method **startPrintJob**. Pages are numbered *0* through *(n-1)*. Each time a page is printed the printer advanced by one page in the document. The print job is "circular", in the sense that after page *(n-1)* is printed the printer begins printing another copy of the document starting at page *0*.

The basic operation of the printer is provided by the method **printPage**, whose behavior is:

1. Reducing the sheets available by 1, but never going below 0.
2. Increasing the total page count printed by the printer.
3. Advancing the next page to print by 1 or resetting the next page to 0 if all pages of document has been printed.

At any time, the paper tray may be removed from the printer. Doing so makes the sheets available to the printer zero. The sheets available may also be zero when the sheets in the tray run out. When this is the case **printPage** should print no pages and should not advance to the next page.

Please note that you should not use any conditional statements (i.e. "if" statements), or anything else we haven't covered, for this assignment. There will be a couple of places where you need to choose the larger or smaller of two numbers, which can be done with the methods `Math.max()` or `Math.min()`. For the wrapping behavior of the print job, you can use the mod (`%`) operator. To keep pages from printing when the tray is removed, just set the sheets available to 0. (*You may be penalized slightly for using conditional statements, because you are just being lazy and making your code longer and more complicated.*)

Specification

The specification for this assignment includes this pdf along with any "official" clarifications announced on Canvas.

There is one public constructor:

```
public Printer(int trayCapacity)
```

Constructs a new printer with the given maximum tray capacity of the number of paper sheets it can hold. Initially the tray is empty and the printer has not printed any pages.

There are the following public methods:

```
public void startPrintJob(int documentPages)
```

Starts a new print job to make copies of a document that is a specified page length (`documentPages`). Updates the next page to print as page 0 (denotes the first page of the document).

```
public int getSheetsAvailable()
```

Returns the number of sheets available for printing.

```
public int getNextPage()
```

Returns the next page number of the document that will be printed.

```
public int getTotalPages()
```

Returns the count of all pages printed by the printer since its construction.

```
public void printPage()
```

Simulates the printer printing a page. The number pages printed is either one or zero depending on whether there is at least one sheet of paper available to the printer. Increments the total page count of the printer by the number of pages printed. Advances the next page to print by the number of pages printed (possibly wrapping around to page zero). The number of pages available to the printer and in the tray are also updated accordingly.

```
public void removeTray()
```

Removes the paper tray from the printer; that is, makes the sheets available to the printer zero.

```
public void replaceTray()
```

Replaces the tray in the printer; that is, makes the sheets available to the printer the same as the number of sheets in the tray.

```
public void addPaper(int sheets)
```

Simulates removing the tray, adding the given number of sheets (up to the maximum capacity of the tray), and replacing the tray in the printer.

```
public void removePaper(int sheets)
```

Simulates removing the tray, removing the given number of sheets (but not allowing the sheets to go below zero), and replacing the tray in the printer.

Where's the main() method??

There isn't one! Like most Java classes, this isn't a complete program and you can't "run" it by itself. It's just a single class, that is, the definition for a type of object that might be part of a larger system. To try out your class, you can write a test class with a main method like the examples below in the getting started section.

There is also a specchecker (see below) that will perform a lot of functional tests, but when you are developing and debugging your code at first you'll always want to have some simple test cases of your own, as in the getting started section below.

Suggestions for getting started

Smart developers don't try to write all the code and then try to find dozens of errors all at once; they work ***incrementally*** *and test every new feature as it's written. Since this is our first assignment, here is an example of some incremental steps you could take in writing this class.*

0. Be sure you have done and understood Lab 2.

1. Create a new, empty project and then add a package called `hw1`. ***Be sure to choose "Don't Create" at the dialog that asks whether you want to create module-info.java***.

2. Create the `Printer` class in the `hw1` package and put in stubs for all the required methods, the constructor, and the required constant. Remember that everything listed is declared `public`. For methods that are required to return a value, just put in a "dummy" return statement that returns zero or false. There should be no compile errors. ***DO NOT COPY AND PASTE from this pdf!*** *This leads to insidious bugs caused by invisible characters that are sometimes generated by sophisticated document formats.)*

3. Briefly javadoc the class, constructor and methods. This is a required part of the assignment anyway, and doing it now will help clarify for you what each method is supposed to do before you begin the actual implementation. (Copying phrases from the method descriptions here or in the Javadoc is perfectly acceptable; however, ***DO NOT COPY AND PASTE from this pdf!*** *This leads to insidious bugs caused by invisible characters that are sometimes generated by sophisticated document formats.)*

4. Look at each method. Mentally classify it as either an *accessor* (returns some information without modifying the object) or a *mutator* (modifies the object, usually returning `void`). The accessors will give you a lot of hints about what instance variables you need.

5. You could start by deciding how to keep track of the number of sheets available. The presence of a method `getSheetsAvailable` suggests we might need an instance variable whose value is the number of sheets available. The method `addPaper` should add to the sheets available. Write a simple test to see if this much is working as it should:

```
public static void main(String[] args)
{
    // create a printer
    Printer p = new Printer(100);
    System.out.println(p.getSheetsAvailable());
    System.out.println("Expected 0");
    p.addPaper(1);
    System.out.println(p.getSheetsAvailable());
    System.out.println("Expected 1");
}
```

(*Tip:* you can find code for the simple test case above, along with the others discussed in this section, in the class **SimpleTests.java** linked from the assignment page on Canvas.)

6. The printer simulates printing pages of a document. The number is pages in the document is provided as an argument when calling **startPrintJob**. This method should update instance variables to store the number of pages in the document and set the next page to print to 0 (the first page of the document). The method **getNextPage** can be used to check that the next page is initially 0. Try a test like this:

```
Printer p = new Printer(100);
p.startPrintJob(5);
System.out.println(p.getNextPage());
System.out.println("Expected 0");
```

7. Next, you might start implementing the **printPage** method which simulates printing a single page of the document. It performs multiple actions, one of which is to increment the next page to print. If a copy of the document has completely printed then the next page to print should wrap around to 0. The total number of pages printed by the printer should also be incremented. This can be tested with the methods **getNextPage** and **getTotalPages**.

```
Printer p = new Printer(100);
p.addPaper(1);
p.startPrintJob(5);
p.printPage();
System.out.println(p.getNextPage());
System.out.println("Expected 1");
System.out.println(p.getTotalPages());
System.out.println("Expected 1");
```

8. Printing can only happen when there is paper available. The method **getSheetsAvailable** implies an instance variable to indicate the number of sheets that can currently be used to print. When the paper tray is removed from the printer, the number of sheets available should to be set to 0 even though the tray still has sheet in it. This implies we need two separate instance variables: one to store the number of sheets available and one to store the number of sheets in the tray. Make sure that **addPaper** does not increase the sheets available or the sheets in the tray beyond the tray capacity. Modify **printPage** to only print when there is a sheet available. In other words, printPage prints either one or zero pages. Try a test like this:

```
Printer p = new Printer(100);
p.addPaper (1);
p.startPrintJob(5);
p.printPage();

// try to print a page with no paper available
p.printPage();
System.out.println(p.getTotalPages());
System.out.println("Expected 1");
```

```
        // add paper and try to print again
        p.addPaper(200);
        System.out.println(p.getSheetsAvailable());
        System.out.println("Expected 100");
        p.printPage();
        System.out.println(p.getTotalPages());
        System.out.println("Expected 2");
        System.out.println(p.getNextPage());
        System.out.println("Expected 2");
```

9. The final three methods to implement are **`removeTray`**, **`replaceTray`** and **`removePaper`**. The **`removeTray`** and **`replaceTray`** methods change the number of sheets available but do not change the number of sheets in the tray. The **`removePaper`** method takes paper out of the tray and simulates placing the tray back in the printer (updating the number of sheets available). The **`addPaper`** method also simulates placing the tray back in the printer (updating the number of sheets available).

10. At some point, download the SpecChecker, import it into your project as you did in lab 1 and run it. *Always start reading error messages from the top.* If you have a missing or extra public method, if the method names or declarations are incorrect, or if something is really wrong like the class having the incorrect name or package, any such errors will appear *first* in the output and will usually say "Class does not conform to specification." **Always fix these first.**

The SpecChecker

You can find the SpecChecker on Canvas. Import and run the SpecChecker just as you practiced in Lab 1. It will run a number of functional tests and then bring up a dialog offering to create a zip file to submit. Remember that error messages will appear in the *console* output. There are many test cases so there may be an overwhelming number of error messages. ***Always start reading the errors at the top and make incremental corrections in the code to fix them.*** When you are happy with your results, click "Yes" at the dialog to create the zip file. See the document “SpecChecker HOWTO”, link #10 on our Canvas front page, if you are not sure what to do.

More about grading

This is a "regular" assignment so we are going to read your code. Your score will be based partly (about a third) on the specchecker's functional tests and partly on the TA's assessment of the quality of your code. This means you can get partial credit even if you have errors, and it also means that even if you pass all the specchecker tests you can still lose points. Are you doing things in a simple and direct way that makes sense? Are you defining redundant instance variables? Are you using a conditional statement when you could just be using `Math.min` ? Are you using a loop for something that can be done with integer division? Some specific criteria that are important for this assignment are:

• Use instance variables only for the “permanent” state of the object, use local variables for temporary calculations within methods.

o You will lose points for having unnecessary instance variables

o All instance variables should be `private`.
• **Accessor methods should not modify instance variables**.

See the "Style and documentation" section below for additional guidelines.

Style and documentation

Roughly 15% of the points will be for documentation and code style. Here are some general requirements and guidelines:

• **Each class, method, constructor and instance variable, whether public or private, must have a meaningful javadoc comment**. The javadoc for the class itself can be very brief, but must include the `@author` tag. The javadoc for methods must include `@param` and `@return` tags as appropriate.

o Try to briefly state what each method does in your own words. However, there is no rule against copying the descriptions from the online documentation. *However: **do not literally copy and paste from this pdf**! This leads to all kinds of weird bugs due to the potential for sophisticated document formats like Word and pdf to contain invisible characters.*

o Run the javadoc tool and see what your documentation looks like! (You do not have to turn in the generated html, but at least it provides some satisfaction :)

- All variable names must be meaningful (i.e., named for the value they store).
- Your code should **not** be producing console output. You may add `println` statements

  when debugging, but you need to remove them before submitting the code.

- Internal (//-style) comments are normally used inside of method bodies to explain *how* something works, while the Javadoc comments explain *what* a method does. (A good rule of thumb is: if you had to think for a few minutes to figure out how something works, you should probably include an internal comment explaining how it works.)

o Internal comments always *precede* the code they describe and are indented to the same level. In a simple homework like this one, as long as your code is straightforward and you use meaningful variable names, your code will probably not need any internal comments.

• Use a consistent style for indentation and formatting.
o Note that you can set up Eclipse with the formatting style you prefer and then use Ctrl-

Shift-F to format your code. To play with the formatting preferences, go to Window- >Preferences->Java->Code Style->Formatter and click the New button to create your own "profile" for formatting.

If you have questions

For questions, please see the Piazza Q & A pages and click on the folder **`hw1`**. If you don't find your question answered, then create a new post with your question. Try to state the question or topic clearly in the title of your post, and attach the tag **`hw1`**. *But remember, do not post any source code for the classes that are to be turned in.* It is fine to post source code for general Java examples that are not being turned in. (In the Piazza editor, use the button labeled "pre" to have Java code formatted the way you typed it.)

If you have a question that absolutely cannot be asked without showing part of your source code, make the post "private" so that only the instructors and TAs can see it. Be sure you have stated a specific question; vague requests of the form "read all my code and tell me what's wrong with it" will generally be ignored.

Of course, the instructors and TAs are always available to help you. See the Office Hours section of the syllabus to find a time that is convenient for you. We do our best to answer every question carefully, short of actually writing your code for you, but it would be unfair for the staff to fully review your assignment in detail before it is turned in.

Any announcements from the instructors on Canvas that are labeled "Official Clarification" are considered to be part of the spec, and you may lose points if you ignore them. Such posts will

always be placed in the Announcements section of Canvas. (We promise that no official clarifications will be posted within 24 hours of the due date.)

What to turn in

Note: You will need to complete the "Academic Dishonesty policy questionnaire," found on the Assignments page on Canvas, before the submission link will be visible to you.

Please submit, on Canvas, the zip file that is created by the SpecChecker. The file will be named **`SUBMIT_THIS_hw1.zip`**. and it will be located in whatever directory you selected when you ran the SpecChecker. It should contain one directory, **`hw1`**, which in turn contains one file, **`Printer.java`**. Please **LOOK** at the file you upload and make sure it is the right one!

Submit the zip file to Canvas using the Assignment 1 submission link and **VERIFY** that your submission was successful. If you are not sure how to do this, see the document "Assignment Submission HOWTO", link #9 on our Canvas front page.

We recommend that you submit the zip file as created by the specchecker. If necessary for some reason, you can create a zip file yourself. The zip file must contain the directory **hw1**, which in turn should contain the files **Printer.java**. You can accomplish this by zipping up the **src** directory of your project. **Do not zip up the entire project**. The file must be a zip file, so be sure you are using the Windows or Mac zip utility, and NOT a third-party installation of WinRAR, 7-zip, or Winzip.