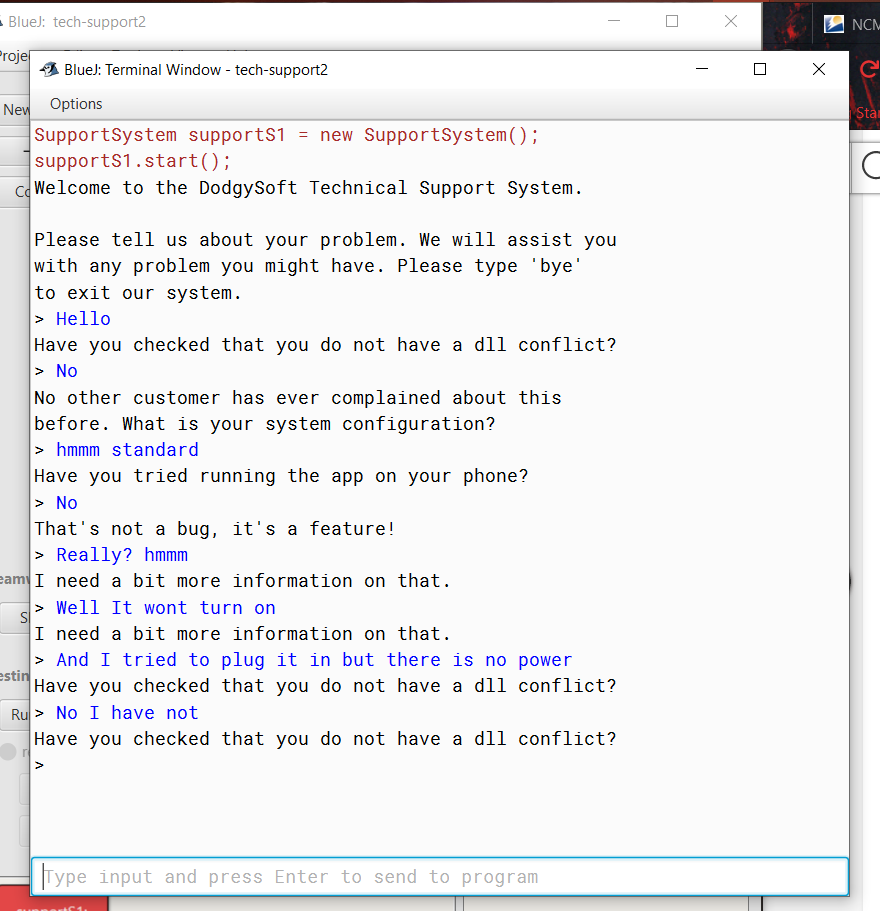
6.1



6.2

Each of the class documentation files is set up with a summary of the fields and the constructors first. Then it also includes a summary of the methods. Each of these sections is made up with one to two columns for information. It also includes sections for details on each of the summary sections with a section for inheritance of the methods as well.

6.3

StartsWith is a method of String that has two loads of execution one takes in as String prefix and an int tooffset. And the other just takes String prefix. What this method does is returns a Boolean true or false depending on weather a string starts with a specified prefix ‘prefix’. The first one that also has an int as a parameter checks only after a specific index number while the other checks the string in its entirety.

6.4

Yes, it is called endsWith. It takes one parameter a String suffix. It returns a Boolean weather or not the suffix supplied is in the String.

6.5

Yes, length returns the length of the character array that is a String. Length takes no parameters.

6.6

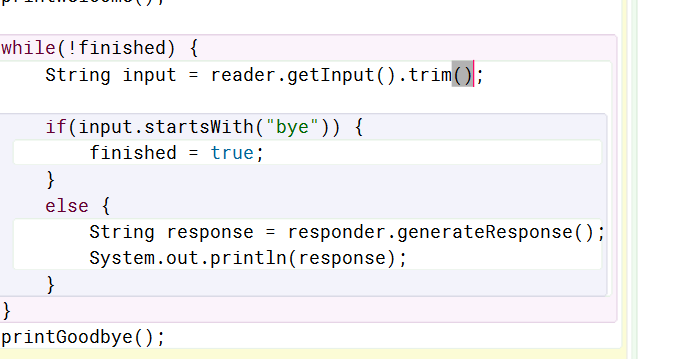
I just went through all the methods in the list. It was pretty easy they are somewhat organized in alphabetical sections that I found easy to follow.

6.7

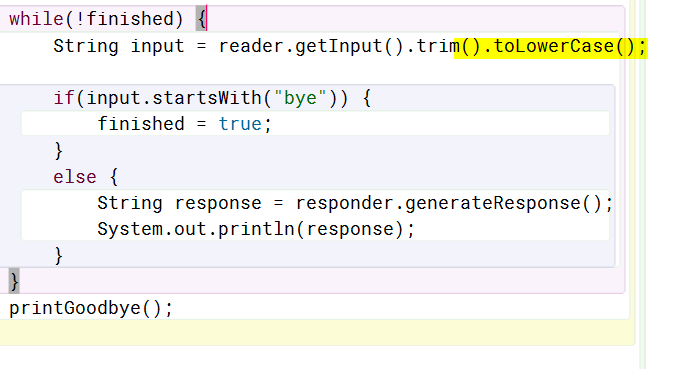
public [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) trim();

Text = text.trim();

6.8



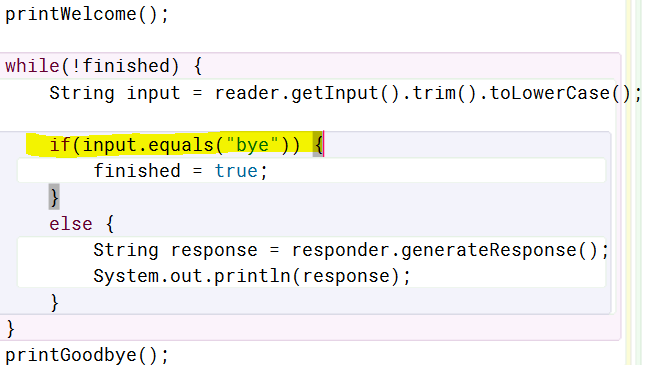
6.9



6.10

Equils returns true and takes an object as a parameter. The method only returns true if the object put in as a parameter is a String object that is equal to the String it is compared to.

6.11



6.12

Random is in the java.util package. Random is used to create Random objects which are random number generators that can be used to set a random value to a variable using one of its methods depending on what random return type you want. You can also add parameters in order to limit the results ie. (2) = 0-1.

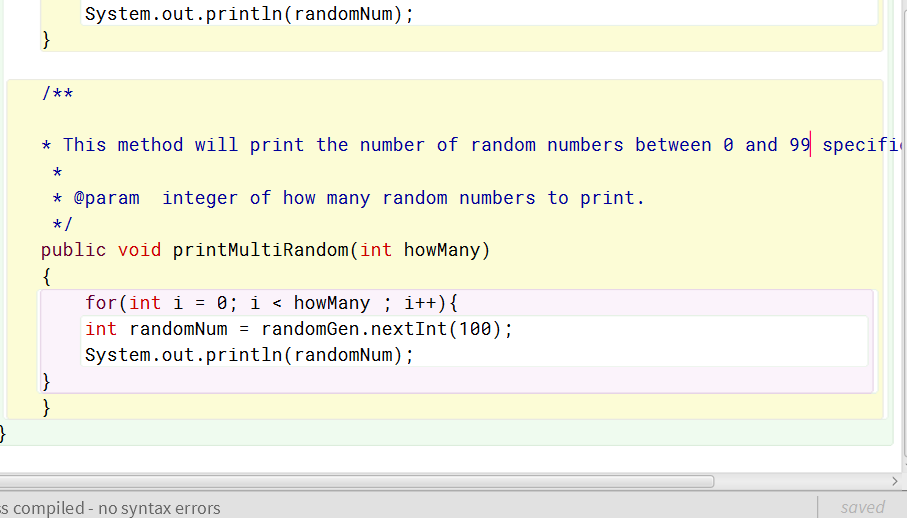
6.13

Random rand = new Random();

int randInt = rand.nextInt();

6.14

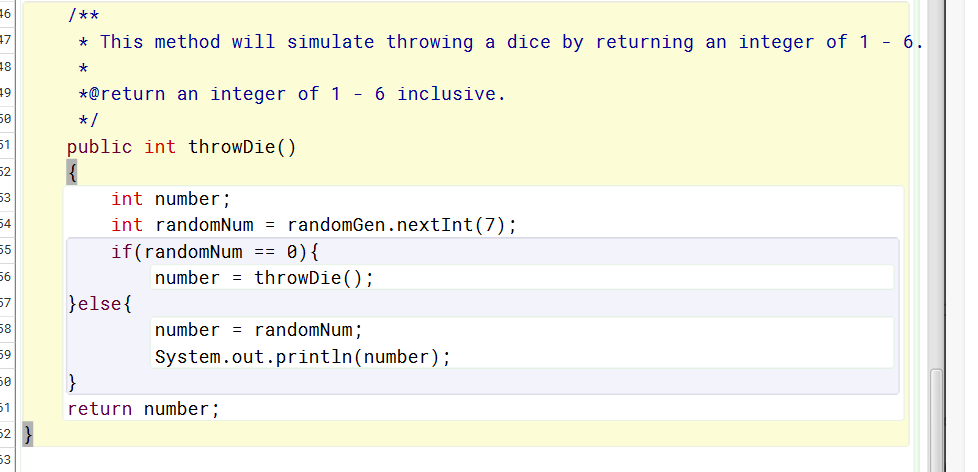




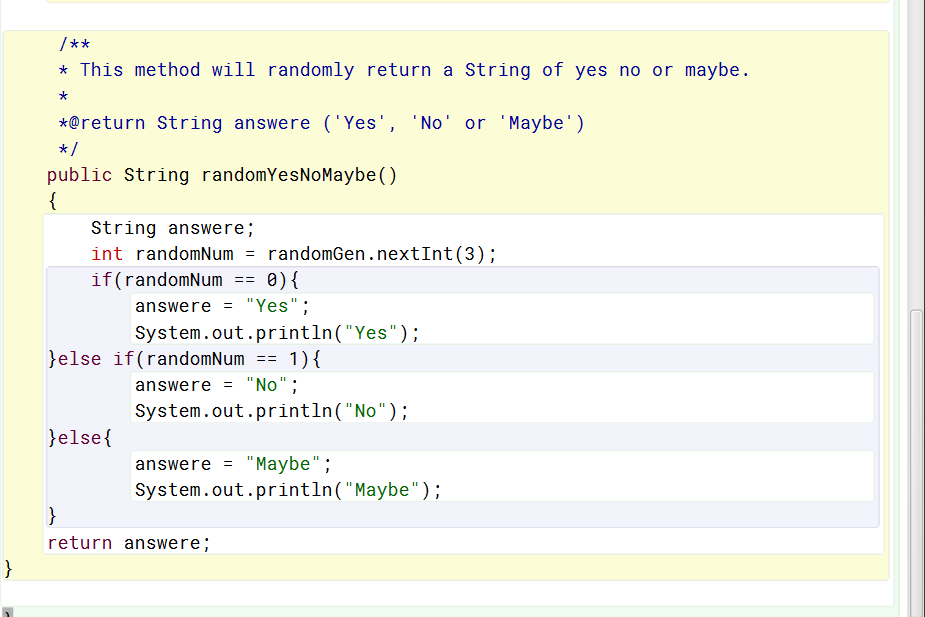
6.15

The possible random numbers generated by randomGen.nextInt(100) would be zero through 99 inclusive and with zero include 100 possibilities.

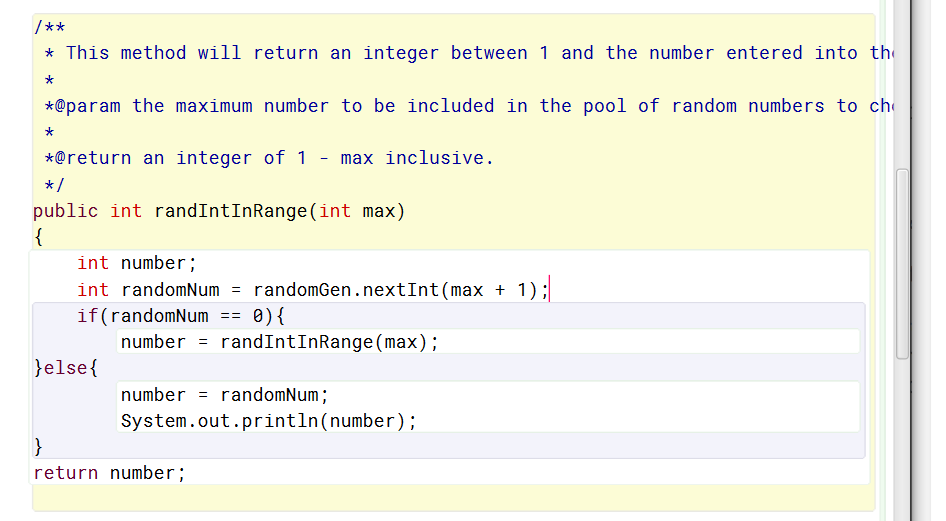
6.16



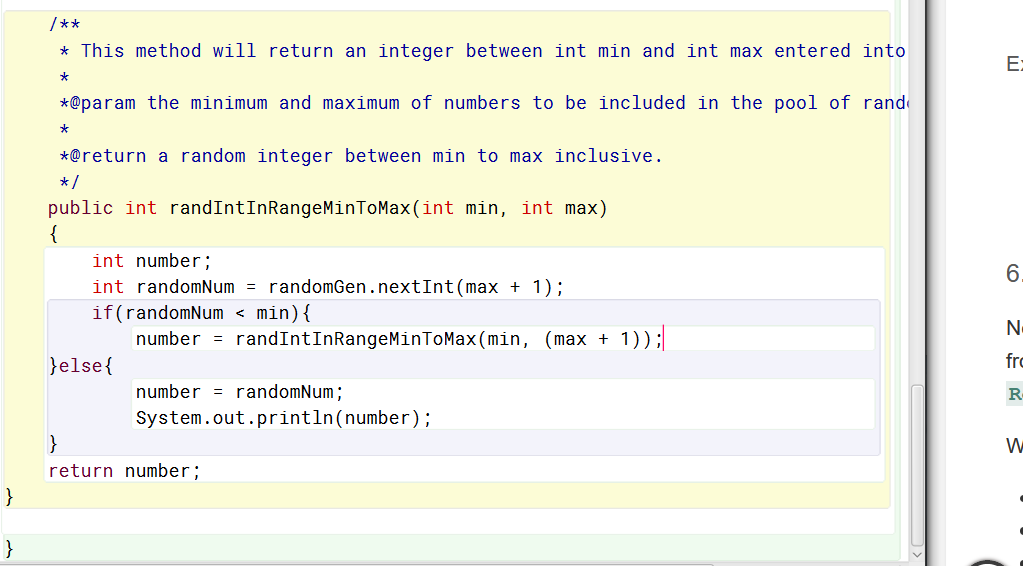
6.17



6.19



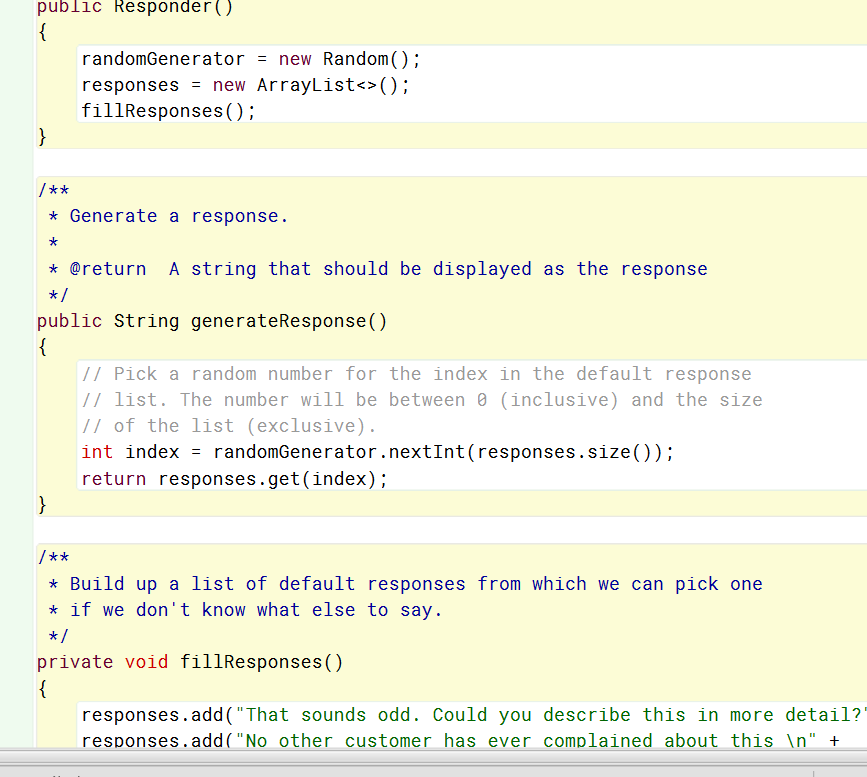
6.20



6.21

The real difference is the seed for the random number generation is non-deterministic in the secure version. This means that the seed is completely unpredictable. This is up to the FIPS 140-2, Security Requirements for Crytographoc Modules. This also means that all output sequences created by SecureRandom are to be cryptographically strong as described in RFC 1750: Randomness Recommendations for Security. You would use the SecureRandom class in situations where you need to be sure that someone cannot determine the seed and in turn predict the outcome. Like in the case of resetting passwords.

6.22



6.23

Yes, it will. Since the responds generator add the responds to a list and uses the length of the list as the limit for the random umber generator. You could add as many responses as you want into the list.

6.24

HashMaps are the same as maps in the way that they are a kind of list that stores key and value pairs. Hash maps differ because they can contain NULLs as well as they are not synchronized. Hash maps do not guarantee that order will remain constant throughout the life of the hash map. Hash maps has two parameters that can affect their abilities. The load factor and initial capacity. Capacity is the expected maximum number of entries. Then the load factor is how full the hash table can get before the capacity is automatically increased.

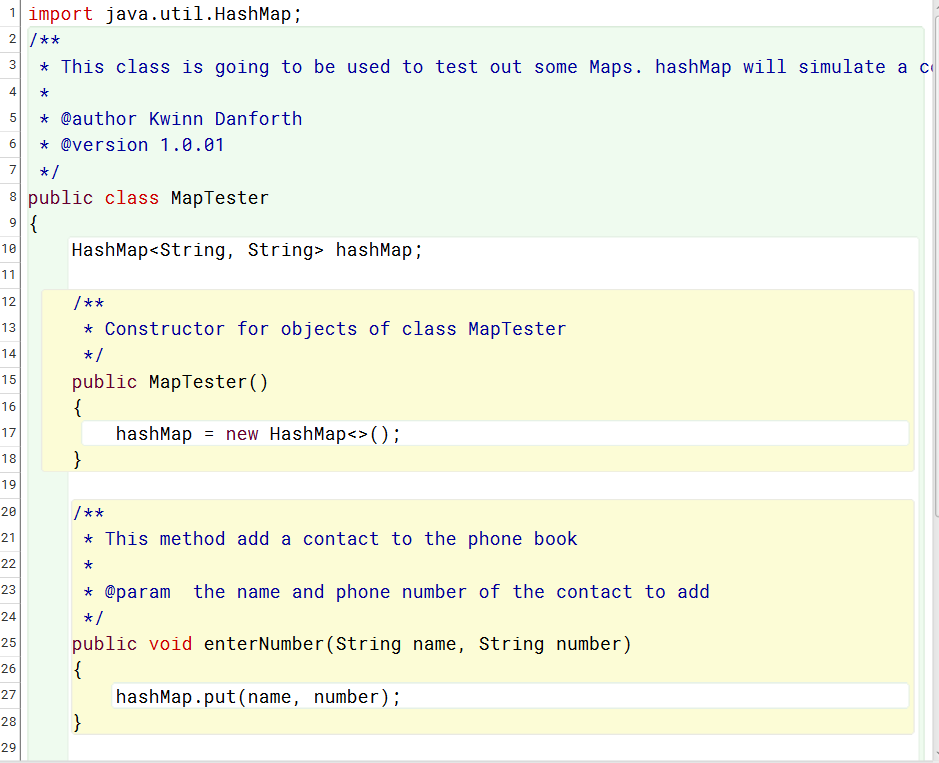
6.25

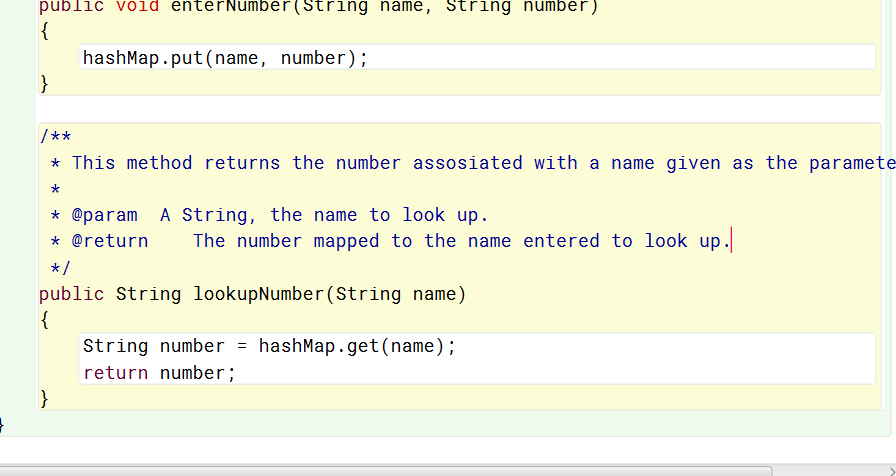
The load factor and initial capacity. Capacity is the expected maximum number of entries. Then the load factor is how full the hash table can get before the capacity is automatically increased.

6.26

To check how many entries are included in a map you can call the size method of the particular map object.

6.27





6.28

So, what happens when you try to add another entry into the hashMap with a key that already exist in said Map. The results are not what you would really expect. What happens is that the new entry essentially overwrites the old one without warning. The expectation would be to throw an exception and not allow the overwrite. This is something to remember and be careful with.

6.29

If you enter in two keys with the same value in each one the hashMap acts like you would think and adds them in with no errors.

6.30

To see if a given key is contained in a Map one would simply call the containsKey method like so. AddressBook.containsKey(“Pops”);. This would return true if there were an entry in the address book with the name Pops as the key.

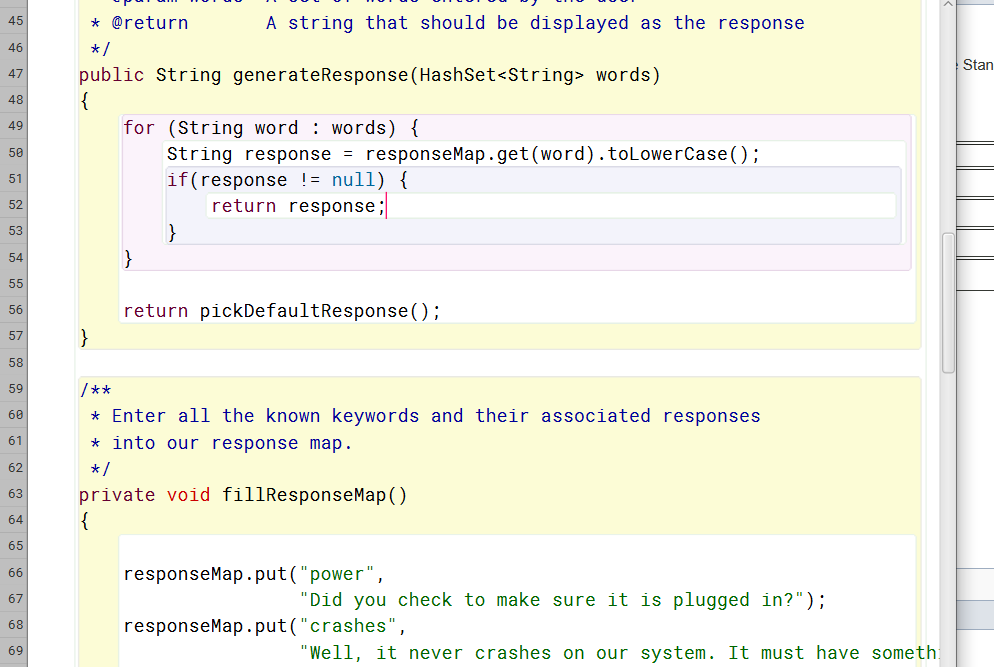
6.31

If you try to look up a key that does not exist in a hashMap than the method returns Null.

6.32

To list all the keys contained in a Map you can use the keySet() method to see all the keys contained in the Map.

6.33



6.34

A set is a set of values with no repeats as is a hash set. The difference is that a hash set is unsorted. The difference between List and ArrayList is that List is an interface and ArrayList is a class.

6.35

The split method can break up strings by supplying a point at which to split given as a parameter. You can do like we did with the example in the book and put in “ “ to break at the spaces. Though you can use any regular expressions to break the charactors up. You can also supply an integer limit to the number of times to match the regex.

6.36

x.split(“\\s”);

x.split(“:”);

6.37

The difference in returning a Hashset when compared to an ArrayList is that a Hashset will return unsorted and an ArrayList will have ordered indices.

6.38

Yes, what seems to happen is that it splits the string up along every space meaning that some of the broke down pieces ended up as only spaces with no characters.

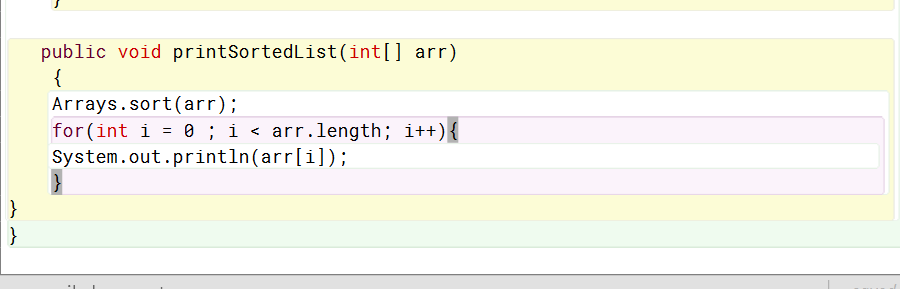
6.39

Arrays.asList is a mehod of the array class. This class can be used to return a list from the elements of an array. It essentially converts an array to a list.

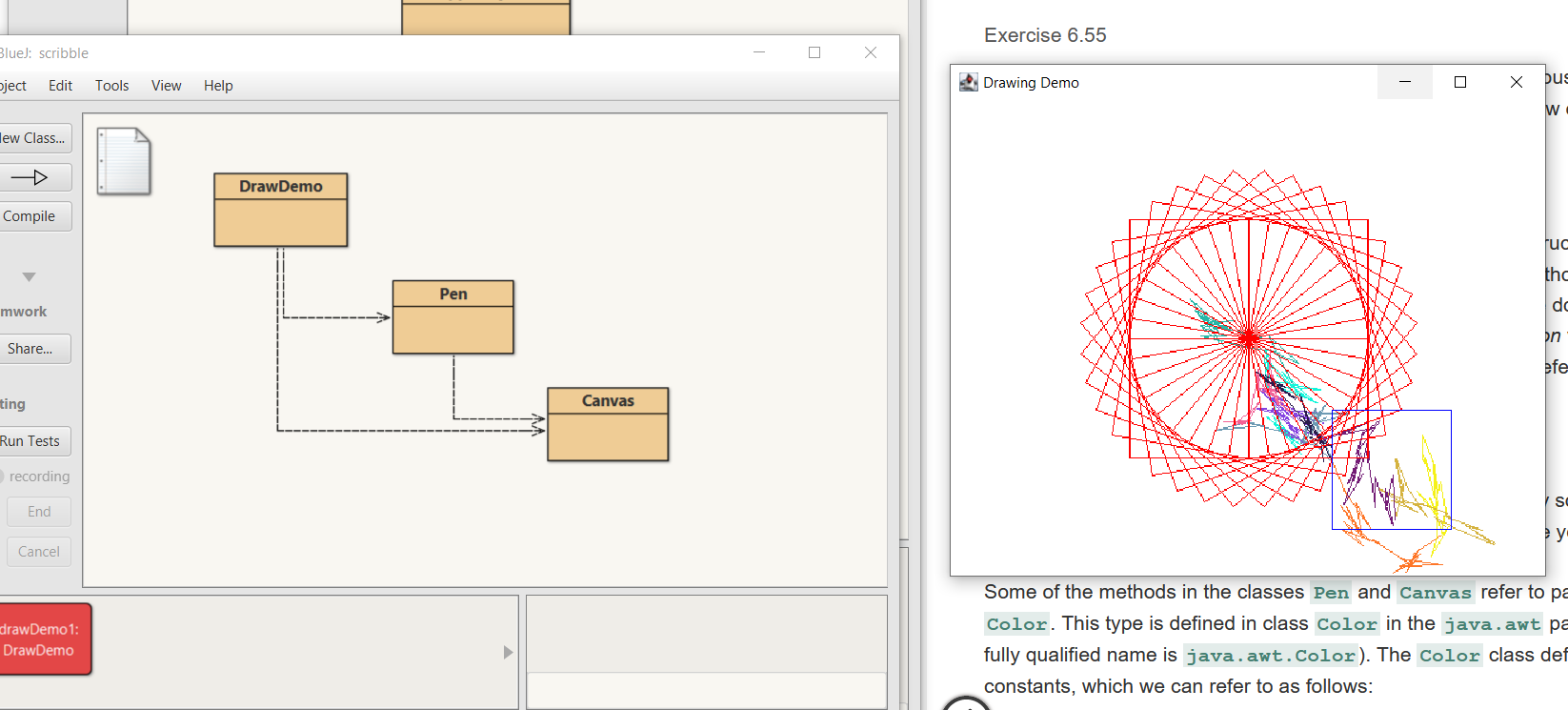
6.40

One example of some more methods of the Arrays class is sort(). Now sort can be very useful. What it does is sorts the specified array into ascending order. Sort can be overloaded to sort arrays 18 different ways depending what parameters are passed to it. Sort could be passed something like two integers for the from and to indexes to sort thing in a specific section of the array.

6.41

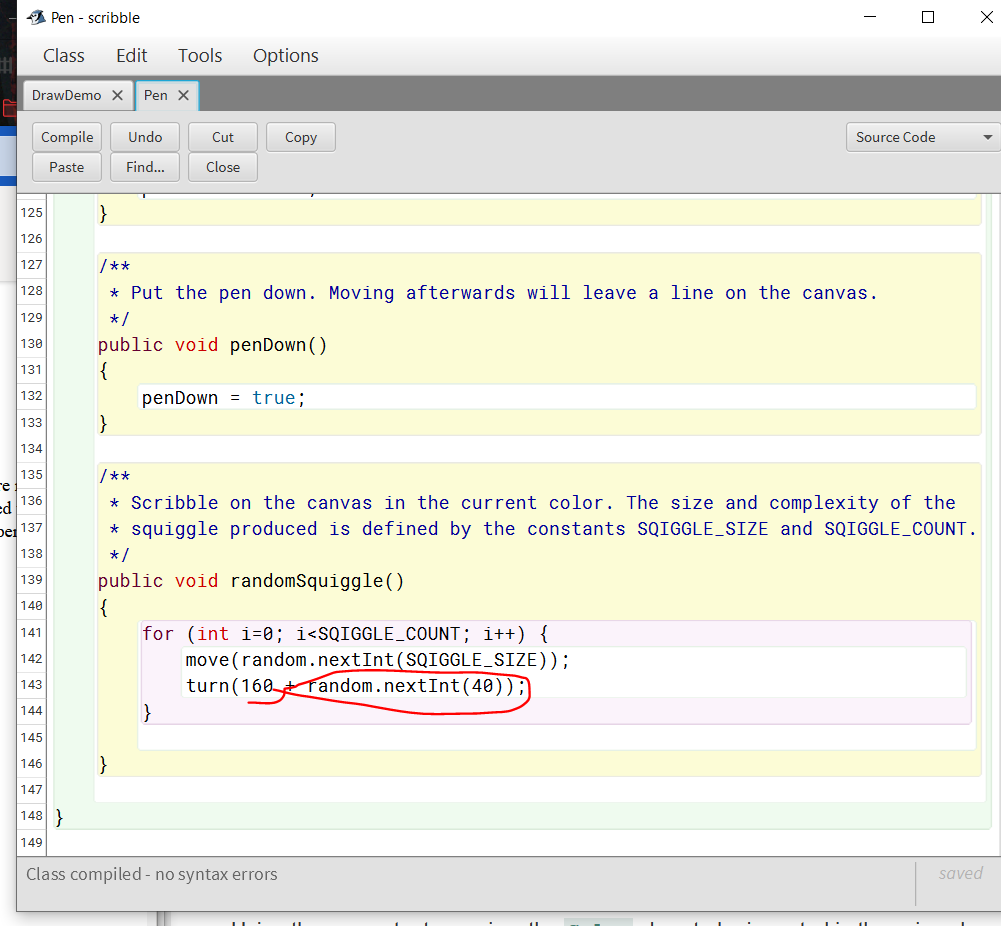
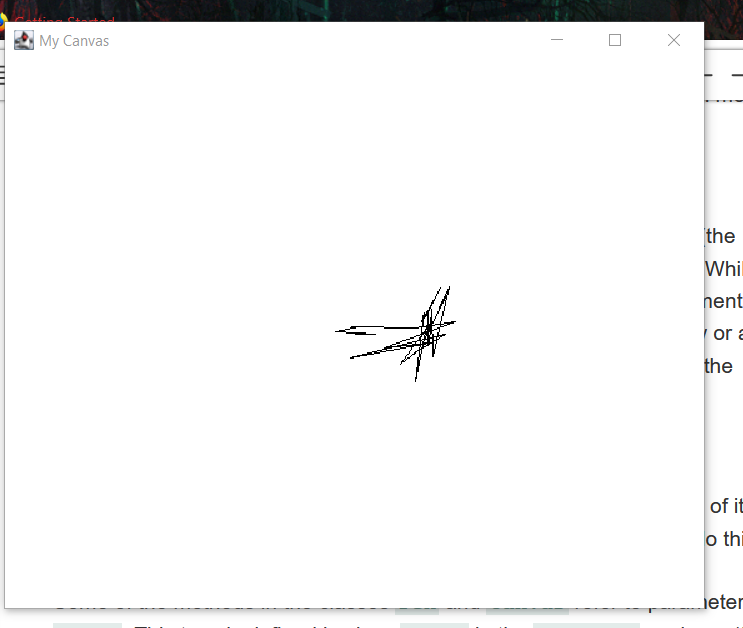


6.55

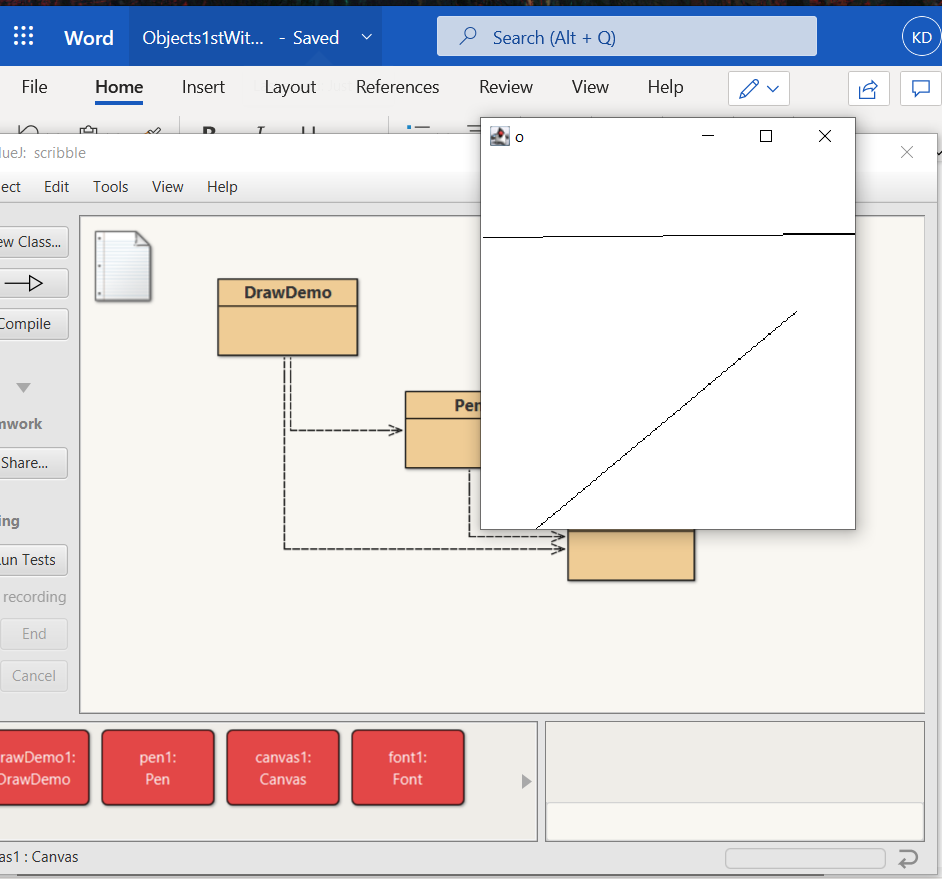


So, the Draw demo class uses a pen in order to draw the square using a turn method. This Square method now is used to create the wheel shape by turning and drawing 36 squares. The squiggle is created using the randonSquiggle() method of the pen class uses some random numbers in order to turn the pen and create random squiggles.

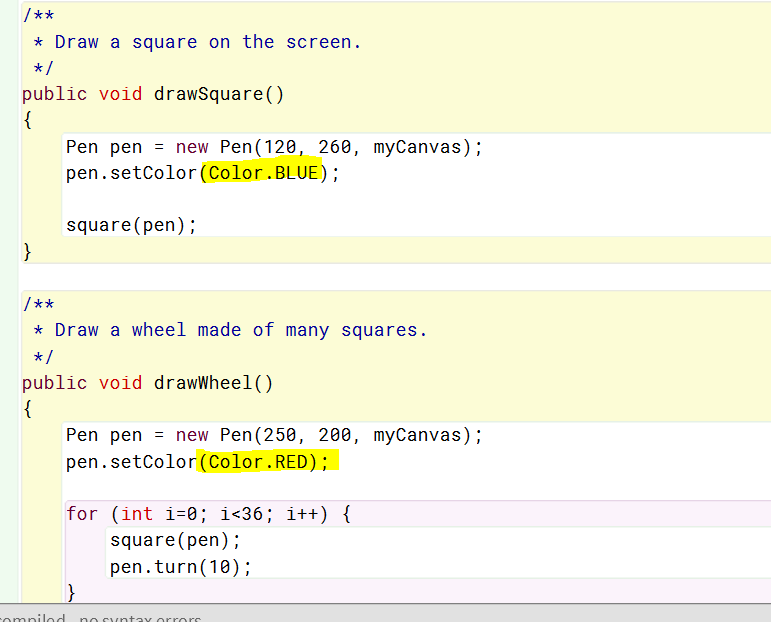
6.56



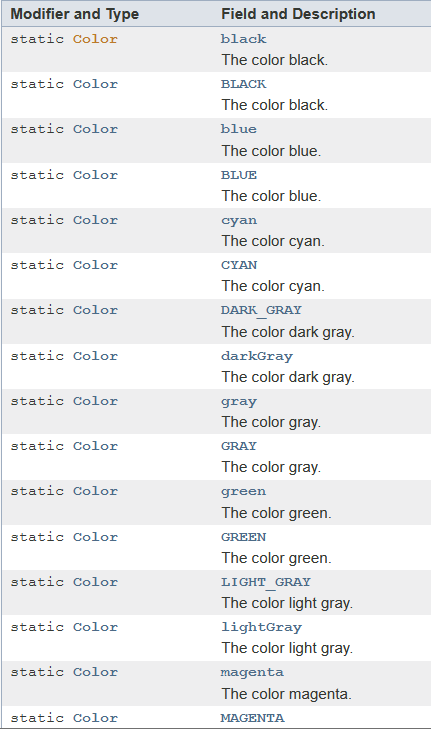
6.57

The canvas class creates a canvas for the pen to draw on. When you interactively create an object of this class you can use to draw lines. As seen in the picture here.  The canvas class has many other methods as well, but I think they are mostly meant to be used by other classes rather than run interactively like this.

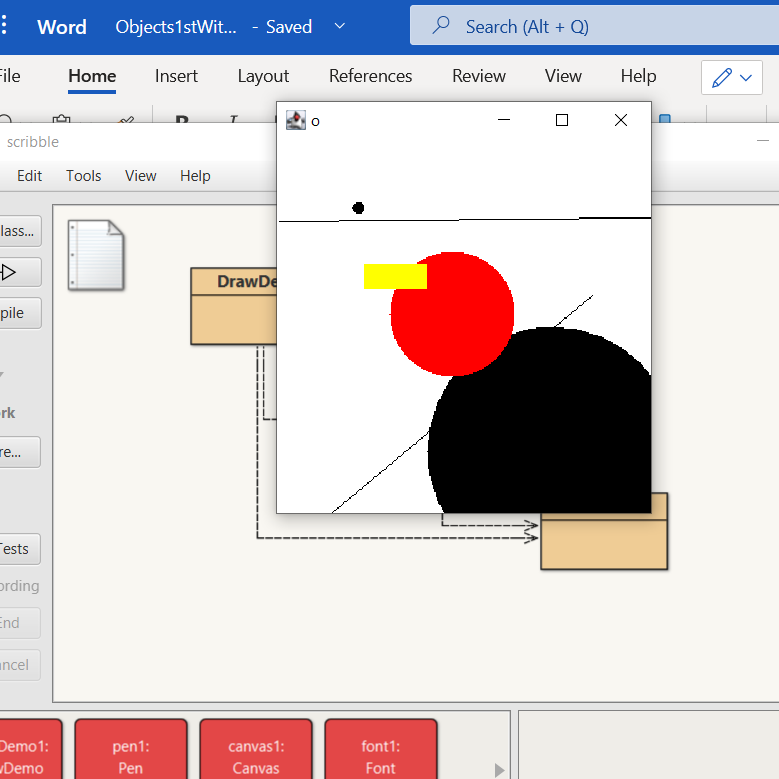
6.58



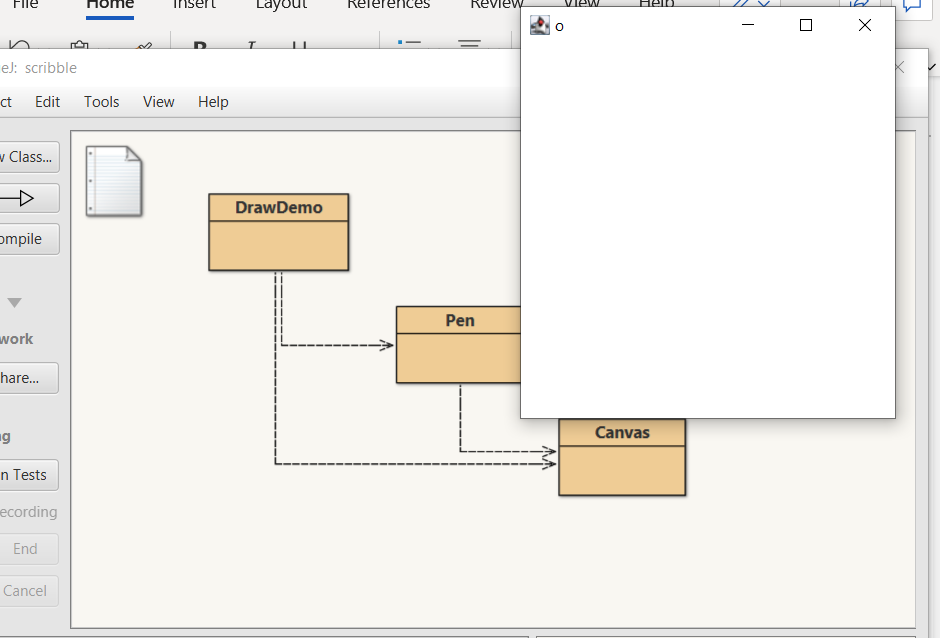
6.59



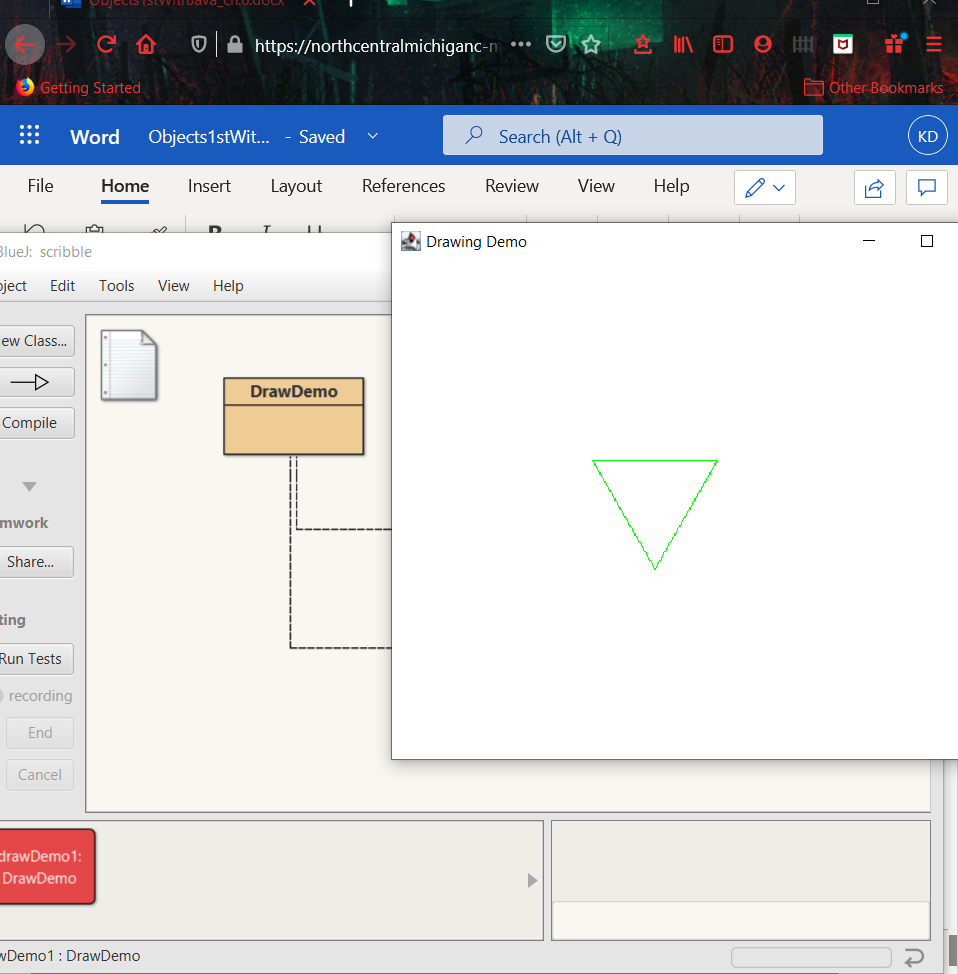
6.60

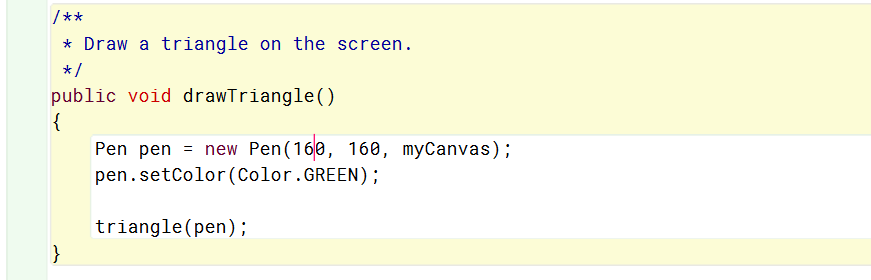


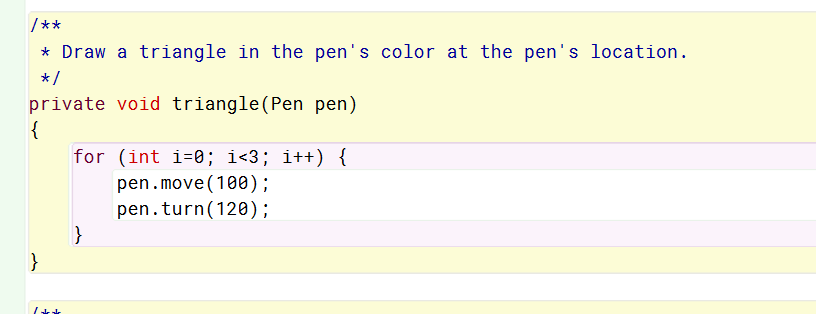
6.61

By calling the earase() method. 

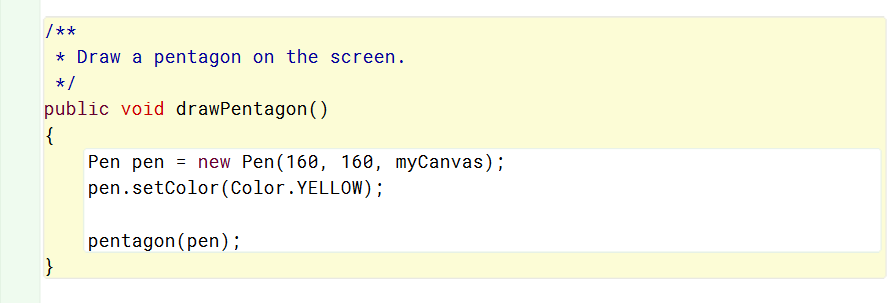
6.62

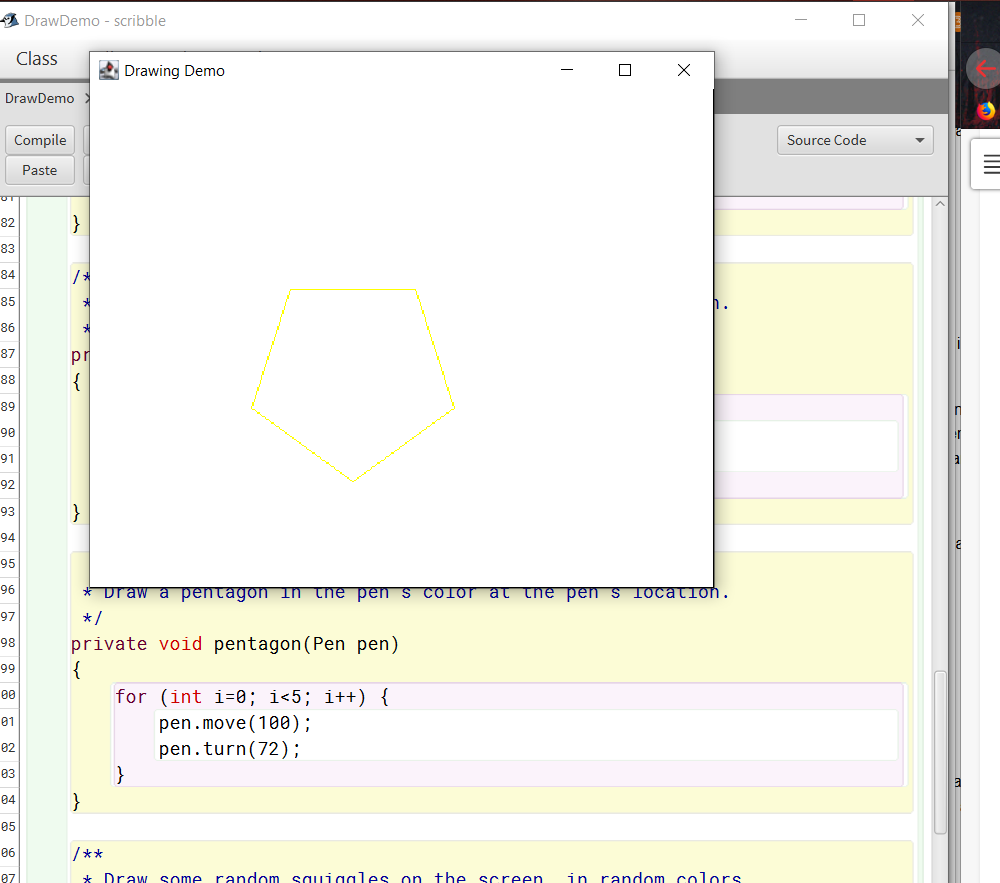




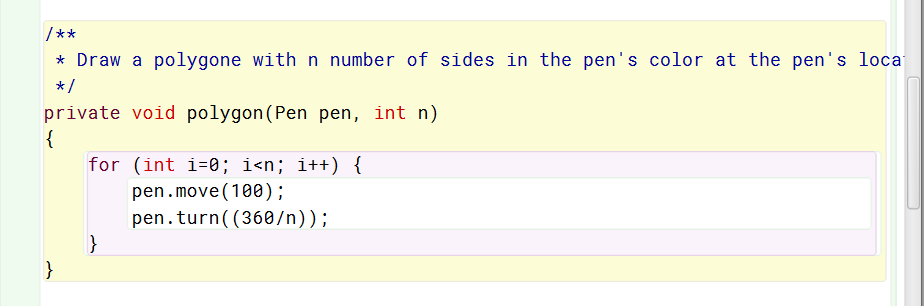
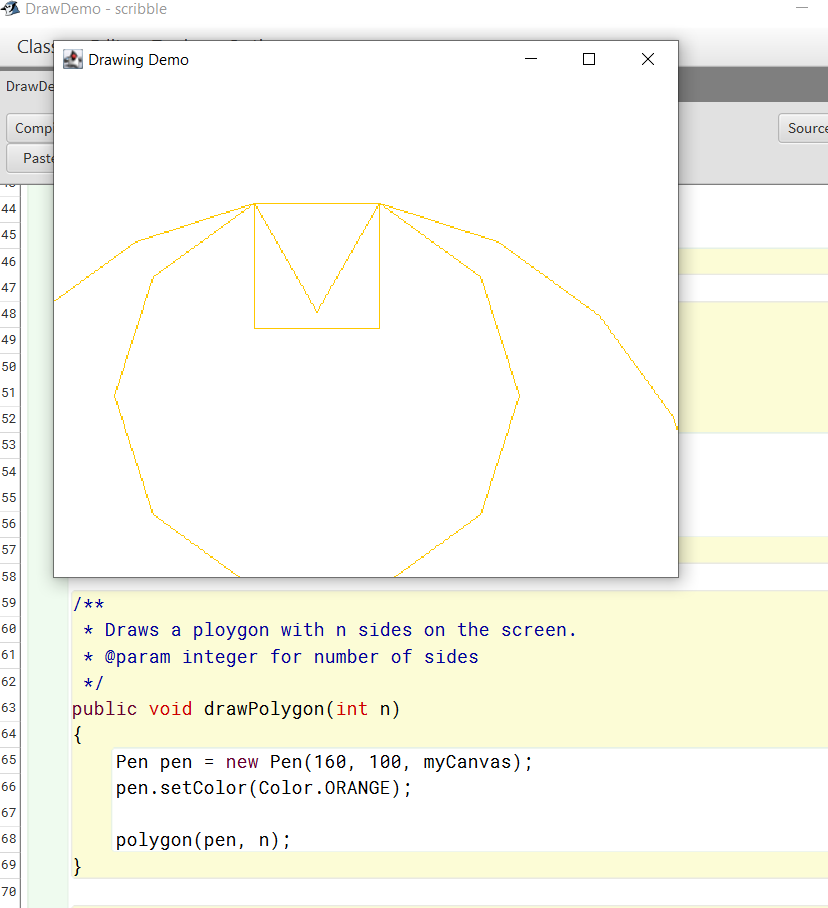


6.63





6.64



6.65

