Use the following outline to guide your self-assessment and notetaking

Week 4 – Loops

The while Statement (Ch 5.4)

How it works

- The code inside the while loop continues to execute until a given condition becomes false
- Once the condition is false, the loop will stop and the program will continue onward

Sentinal value

- This is a value used in a loop with user input to terminate the loop
- Indicates the end of input
    - Ex: the user may enter 0 when inputting a series of numbers to show the end

Infinite loops

- Occurs when the loop condition is never evaluated to false causing the program to run endlessly
- Ctrl-C will terminate the program, so use it if you have an infinite loop

Nested loops

- Loops can exist within loops as well
- Inner loop executes completely for every iteration of the outer loop
- Normally used to break a larger task up into smaller units:
    - Ex: palindrome check with a series of strings:
        - Outer loop iterates over each string
        - Inner loop iterates over each character in a particular string

The break and continue Statements

- Used to transfer the flow of execution to the statement after the one in the current flow
    - Break: exit the loop and necessary when writing switch statements
        - Can cause program flow to jump from one place to another, which is why it is not always good practice to write within a loop
    - Continue: Move on to the next iteration in the loop
        - Also can be avoided in a loop

Iterators (Ch 5.5)

Definition: Any object that has methods that allow you to process a collection of items one at a time

- Ex: Arrays and arraylists

Reading text files

- Uses a scanner to read the lines of the file
- For user input,

    - System.in is passed inside the constructor of the scanner object
- For files,
    - A file object is passed into the scanner. It is initialized with the path to the file
- If there is a problem due to a file not found, or the system cannot open the file, an IO exception is thrown

The ArrayList class (Ch 5.6)

Defining

- ArrayList is part of the java.util package, so you need to import java.util.ArrayList before creating one
- Also can be imported from the collections API
- Does not have a predefined length unlike regulary arrays
- Syntax:
    - Arraylist <Object> [name] = new ArrayList<E>()

Important ArrayList methods

add(E.obj)	Appends the object E to the end of the list
add(int index, E obj)	Inserts the object E after the specified index
clear()	Remove all elements from the list
remove(int index)	Pops the object located at the specified index
get(int index)	Returns the value at the specified index
indexOf(Object obj)	Returns index of the first occurrence of obj
Contains(Object obj)	Checks if list contains the specified object
size()	Returns length(number of elements) in the list
isEmpty()	Checks if list has nothing inside it

The switch Statement (Ch 6.1)

Defining

- Switch works just like multiple if statements to match a value to a particular case
- More readable than multiple ifs
- Syntax:

```
switch(condition)
{
        case 1:
                …
                break;
        case 2:
                …
```

```
            break;
        …
        default:

            …
        }
```

Use of break

- Processing jumps to the statement following the switch block
- Breaks out of each case
- Without break, the processing would continue to the next case in the switch statement

Ch 6.2, 6.3 are optional, but interesting and useful (not tested)

The for Statement (Ch 6.4)

Defining

- The for loop is useful for when you know how many iterations are expected
- Consists of a start, an end, and a step value

Structure

```
for (int x = start; x < end; x += step){
…
}
```

- The range could be defined going forward or backwards as well and you can use any comparison operator in the end part of the loop header

The for-each loop

- This is a more enhanced version of the for loop where the loop variable goes over all elements in an iterable
    - Ex:
        - For (Book myBook: library){

            …

            }

Here, the library is the iterable and the myBook variable is the variable that loops over all the elements

Comparing loops

- While and do loops are used when we do not know how many iterations are expected
    - The only difference between them is when the condition is evaluated
        - Do while loops run at least once before evaluating the condition
        - While loops evaluate the condition before running
- For loops always have a set number of iterations