CS1050 Technical Documentation

Use headings to organize topics and create a table of contents to be able to quickly access information.

[Overview of Developer Technical Documentation](#_e7hnk3gmh67o)

[Set Up Development Environment](#_efvt7vjb7dhk)

[Version Control with Git and Github](#_p4s5uluouzyw)

[IDE Information](#_xda8plfuejto)

[General Resources](#_j8bsu2nohj85)

[Module 1](#_qn332jwlduxf)

[Compilation process](#_8u1ro2aazjxd)

[Variables, Constants, and Primitive Data Types](#_v362y395npbk)

[Arithmetic Operators and Combined Assignment Operators](#_ekcaf49ktnl3)

[Memory Allocation, Primitive Data Types, Conversion and Casting](#_t5qvme324700)

[Quality Code: Naming Conventions](#_2gp23qqtt32h)

[Module 2](#_qke7paye0rm)

[Classes, Objects and Methods](#_41t39wfjzewk)

Follow the 4 tips from [Part 2: Developer Technical Documentation](https://docs.google.com/document/d/1Ve-3OD9EN9DCufeGZTyH_vIt9bsDzUMUv_p8KI1CXlQ/edit#heading=h.4lr6hykc4ng6). This can be done in many ways

* Organize use headings and subheadings
* Summarize, list, screenshots
* Links to resources
* Copy code snippets with comments in your code explaining
* Link to code examples in your repository
* Other resources such as websites and videos. Just remember not all information is good information.

# Overview of Developer Technical Documentation

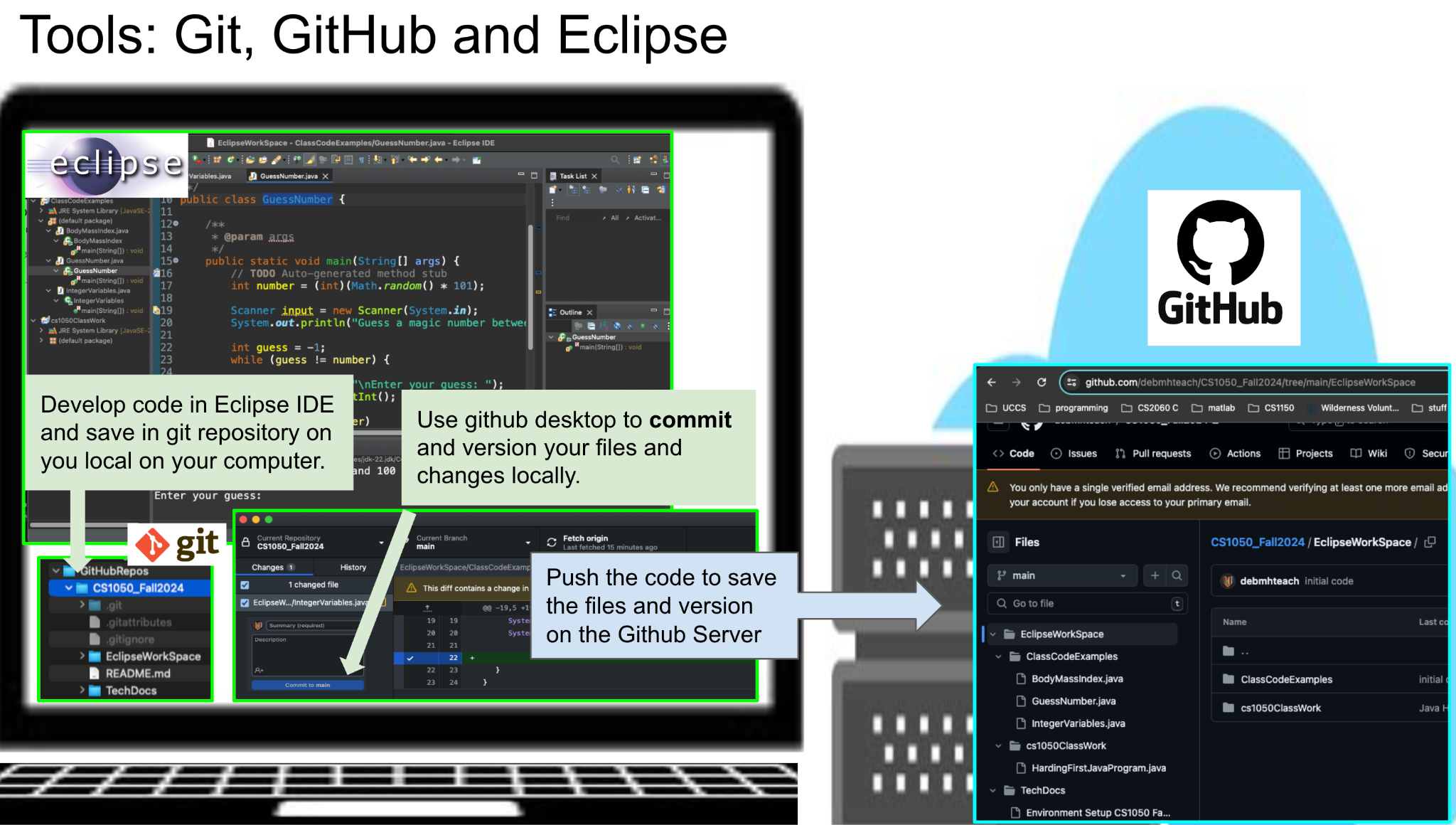
I use headings to help organize and easily access information. I like to include snippets of code that contains comments as an easier way to have documentation. You can put in the document what is helpful for you.

This is your documentation to organize according to your needs. It should include at least the following but can contain more.

# Set Up Development Environment

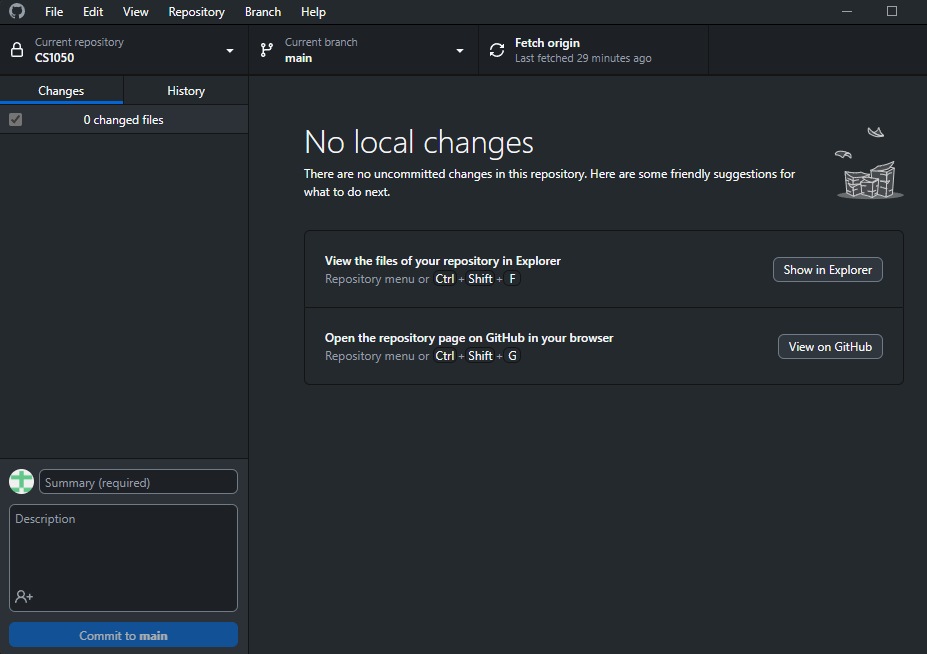
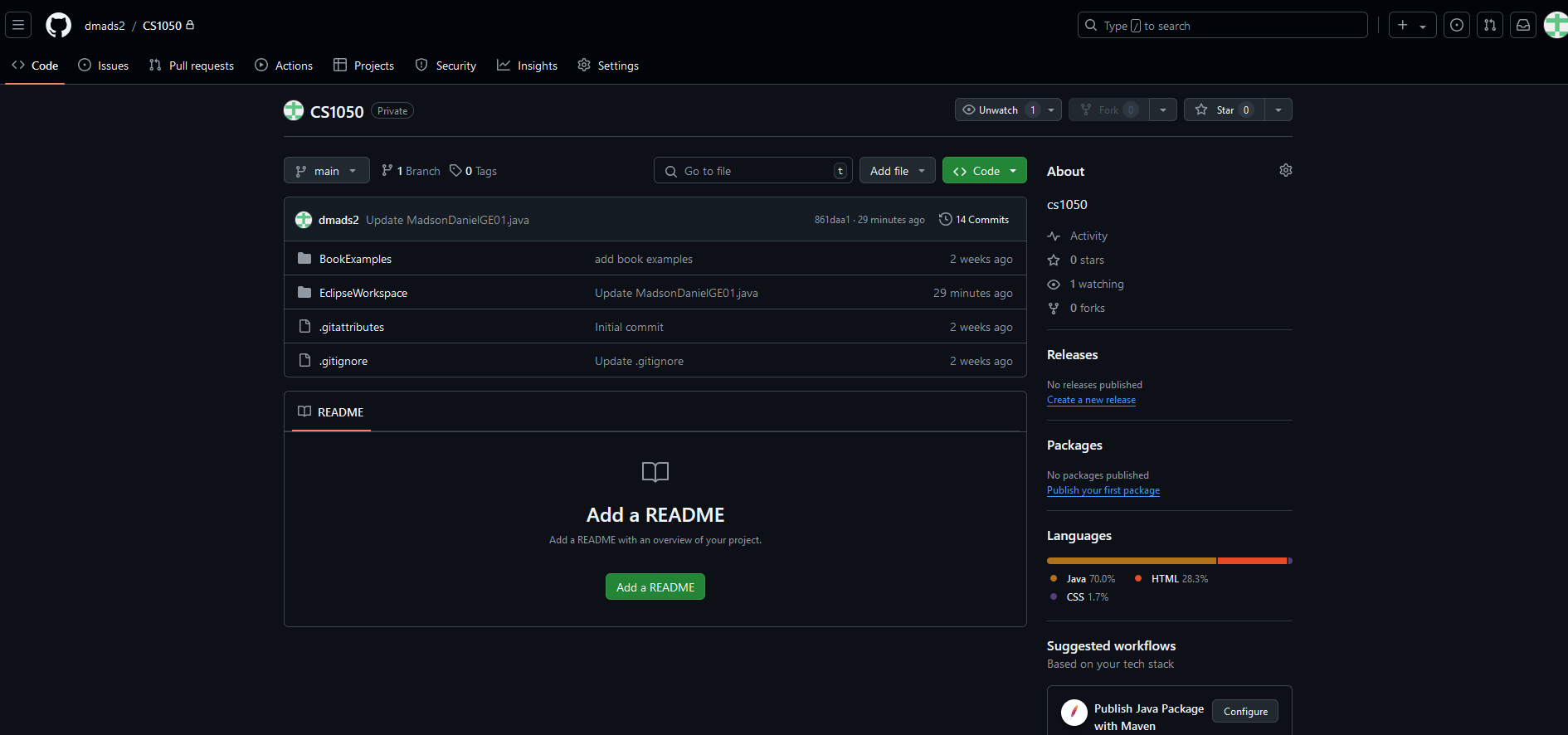
Include what is necessary to set up your environment again. It can be a link to other documents or resources but it is required to have a brief summary of what information is provided in the link.

[Environment Setup CS1050 Fall 2024.docx](https://docs.google.com/document/d/1lnLZdxusq1UNoUshdf3sK64QaTrNF5Iy/edit#heading=h.gjdgxs)



## Version Control with Git and Github

<https://docs.google.com/document/d/1lnLZdxusq1UNoUshdf3sK64QaTrNF5Iy/edit#heading=h.gjdgxs> contains information on how to set up git and github.

* Git-
  + Git is basically a versioning software for your local PC
  + 
* Github-
  + Github is git but online. Some benefits include versioning, online backups, code review, and collaboration.
  + 

## IDE Information

Setting up a project with IDE

<https://docs.google.com/document/d/1lnLZdxusq1UNoUshdf3sK64QaTrNF5Iy/edit#heading=h.gjdgxs> contains information on how to set up Eclipse, create a project and how to create a class..

Is there something you want to put here that you will be doing a lot from the documentation?

<https://dzone.com/articles/how-add-existing-files-eclipse>

HOW TO ADD FILES TO ECLIPSE

CREATING A NEW CLASS

1. Select the appropriate project folder you want the class in
2. Source folder – should be the project name
   1. Package- empty
   2. Name- name for file starts with capital letter
   3. Which method stubs?
      1. Check: public static void main (String[]args)
      2. Uncheck: inherited abstract methods
3. check generate comments
4. done

HOW TO IMPORT A CLASS

1. file > import
2. general > file system
3. click browse directory and select the proper one
4. where it says into folder: select the folder you want the .class and .java file to be saved under
5. finish

# General Resources

Here you can list resources that you use frequently. You can add more.

* [Shared student resources containing resources, lectures and assignments](https://drive.google.com/drive/folders/1HvYY8zzSwlsH--03olvqOJooGnJkZ7F4)
* [Draft Schedule](https://docs.google.com/spreadsheets/d/1igBbmOBTXfvEVicyAggnqRIpV5Fwqh64/edit?gid=2047083326#gid=2047083326)
* [Link to join lecture in teams](https://teams.microsoft.com/l/meetup-join/19%3aklQhREluFbWiaroMMZPBYeNPhZa9AFGnTb7ATIPTUFE1%40thread.tacv2/1724008042961?context=%7b%22Tid%22%3a%2203309ca4-1733-4af9-a73c-f18cc841325c%22%2c%22Oid%22%3a%2233eb6fec-88d5-4bc1-bb67-32063f1cfacc%22%7d)

## 

# Module 1

Rather than typing information

* Copy and paste code examples that contain comments
* Summarize concepts
* Include information form lectures
* Links to tutorials that help you.
* Link to code examples in your repository
* Screenshots
* Other resources such as websites and videos. Just remember not all information is good information.
* Use information from your guided exploration

## Compilation process



* Bug icon- debugger
* Green and white arrow- run



* Termination button- Red square
* Console- where the code gets output

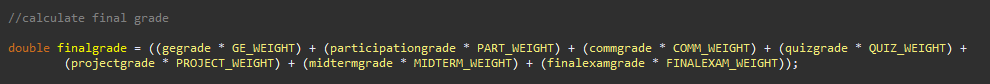
## How to debug

* <https://www.eclipse.org/community/eclipse_newsletter/2017/june/article1.php>
* <https://docs.google.com/document/d/1umJcRMhhoILzl8KGk-3X3ucMHwFFTqMN/edit>
* 1) double click next to the line number you want to pause to step through code
* 2) click the debug button and if asked to switch to debut perspective, check remember my decision
* 3) you will see the statement highlighted to show what statement you will start stepping through
* 3.5) there should be a window to see the variables you are looking at
* 4) use the step over arrow F6 to step through the program line by line

## Variables, Constants, and Primitive Data Types

### 

## Arithmetic Operators and Combined Assignment Operators



* WHEN DIVIDING TWO INTEGERS- YOU WILL GET ANOTHER INTEGER
* You can divide a decimal by an int and it'll be fine
* Cannot divide an int by a decimal

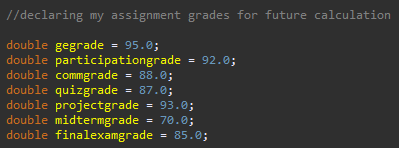
## Memory Allocation, Primitive Data Types, Conversion and Casting

<https://www.javatpoint.com/java-data-types> This website describes primitive data types

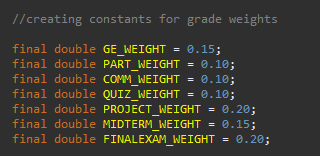
* boolean data type- true/false
* byte data type-
* char data type
* short data type
* int data type- 1,2,3
* long data type
* float data type- 10.0
* double data type- 10.000000000000000000000000000000

## Quality Code: Naming Conventions

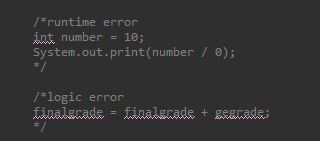
Variables- lowercase and one word



Constants- uppercase and underscores



**Errors:**

****

# Module 2

## Classes, Objects and Methods

## Type casting

* widening (implicit casting)
  + casting a value w/ a smaller range to a bigger range
  + ex- double d = 3;
* narrowing (explicit casting)
  + casting a value w/ a bigger range to a smaller range
  + data can be lost in this process, therefore must be done explicitly
  + ex- int number = (int)3.0;

## Converting data types

* converting int into long (implicit casting)
  + int x = 7;
  + long y = x;
  + therefore y = 7 = x which is a long
* converting double into long (explicit casting)
  + double d = 155.55
  + long k = (long)d;
  + therefore long k = d = 155.55

## Print formatting

* use printf or println
* Println
  + \t new tab, \n new line
  + Ex- System.out.println(“\t Final grade calc”);
* Printf
  + Can be used to format data
  + %b- boolean output
  + %c- a character
  + $d- a decimal integer
  + %f- a floating point number
  + %e- a number in standard scientific notation
  + %s- a string
  + Ie- System.out.printf(“count is %d and amount is %f”, count, amount);
    - Display- count is 5 and amount is 45.560000
  + 
    - This will print out the bmi with one decimal place

## Input from keyboard

* 1) import scanner class
* 2) create new scanner object
* 3) take in input via a new variable
* 4) close input at end of program
* Ex-
  + 1) import java.util.Scanner;
    - (need to do this above the class- ie above public static void main)
  + 2) Scanner input = new Scanner(System.in);
  + 3) double radius = input.nextDouble();
  + 4) input.close();

## Math class- <https://www.javatpoint.com/java-math>

* No need to import- it is in the default java.lang package
* Indexs relation to chars
* A diagram of a number

  Description automatically generated with medium confidence
* There are two constants in the math class
  + PI ~ 3.14…. and E = 2.71828
* Some common methods
  + Includes plenty of methods like Trig, exp() exponent, rounding, min, max, abs (for absolute value), random, log(), log10(), pow(), sqrt()
  + Examples of using these methods
    - Trig method examples
      * Math.sin(double a) (basically asks for a double a which is an angle in radians)
        + double valueSin = Math.sin(Math.Pi/2)
      * Can also do Math.cos() and math.tan()
      * toRadians(degrees) and toDegree(radians)
        + ie- double angleRadians = Math.toRadians(angleDegrees);
* rounding methods
  + ceil()- rounds x up to the nearest integer
  + floor()- rounds x down to the nearest integer
  + rint()- rounds x up to the nearest integer- if equally close to two integers, it returns even one
  + round()- returns int if x is a float or returns long if x is a double rounded to nearest integer value
  + Long round()
* Math.random()
  + Returns a positive double value, greater than or equal to 0.0 and less than 1.0
    - Basically, 0 <= Math.random() < 1
  + Examples
    - double randomDouble = Math.random();
    - int randomInt = (int)Math.random();
      * Basically returns 0 bc its an int
    - randomInt = (int)(Math.random() \* 10);
      * Between 0 and 9
    - randomInt = (int)(Math.random() \*10 +1);
      * Between 0 and 10
  + Creating random integers beteween a and a+b-1
    - a + Math.random() \* b
  + **How to create random numbers between a max and a min**
    - randomMonth = (int)(Math.random() \* (MAX-MIN + 1) + MIN);
      * this would be between 1 and 12 bc months
    - System.out.printf(“Random value for month between %d and %d:\t %d\n”, MIN, MAX, randomMonth);

## Char data type

* Index relations to chars
  + A diagram of a number

    Description automatically generated with medium confidence
* Use single quotes (double quotes for strings)
* A char variable can hold one single character
  + Ex
    - Char middleInitial = ‘M’;
* UNICODE
  + Contains ASCII and much much more
* ASCII Tables- <https://ascii-tables.com/>
* ASCII is 8 bit
* Ex- 01100010 represents the char ‘b’
  + 01100010 = 98 bc (2^7, + 2^6, + 2^5, + 2^3, + 2^3, + 2^2, + 2^1)
    - 0+64+32+0+0+0+1+0 = 98

**Casting chars to numeric values**

* Can be cast to byte, short, int, long, float, double (only includes lower 16 bits)
* In essence- taking a chars unicode and casting it to a numeric value
  + Ex casting char to an int-
    - int x = (int)’b’;
    - System.out.println(“value in int x = “ + x);
    - Displays: value in int x= 98

**Casting numeric values to chars**

* Can be cast to byte, short, int, long, float, double (only includes lower 16 bits)
* Ex casting num to char-
  + char c1= (char)98;
  + System.out.println(“value in c1 = “ + c1);
  + Displays: value in c1 = 98
* Ex 2-
  + Byte number = 98; (98 fits in a byte so this is fine)
  + Char c2 = (char)number;
  + System.out.println(“Value in c2 = “ + c2);
  + Result displays: value in c2 = b

**Reading in chars**

* 1) import scanner class
* 2) create input object to read data
* 3) read in the character
* **Example of printing a char’s ascii value**
  + Scanner input = new Scanner(System.in);
  + System.out.println(“Enter a char”);
  + char someChar = input.next().charAt(0);
    - charAt(0) returns the character at the specified index of whatever was entered by the user
  + System.out.printf(“A letter was entered with ascii value %d \n”, (int) somechar);
* A computer code with text and arrows

  Description automatically generated with medium confidence
* A screen shot of a computer code

  Description automatically generated

**Basic Char methods**

* Character.
  + isDigit(someChar)
    - true/false
  + isLetter(someChar)
    - true/false
  + IsLetterOrDigit(someChar)
    - True/false
  + isLowerCase(someChar) and isUpperCase(someChar)
  + .toLowerCase(someChar) and .toUpperCase()
    - Turns into lowercase or uppercase

**Comparing and testing characters**

* The unicode ascii decimal value is what is used in the comparison
* A computer code with text

  Description automatically generated with medium confidence

**Strings**

* Use double quotes
* How to declare and initialize a string
  + String studentName = “Danny Madson”;
* Strings are not a primitive data type
  + ie- strings can hold many things, not just 1
* **String methods**
  + length()- returns the number of characters in a string
    - ex-
      * int myStringLength = myString.length();
      * System.out.println(“The length of my string is = “ + mystringLength);
  + charAt(index) returns the character at the specified index within a string
    - char D a n n y
    - Index in string 0 1 2 3 4 (d starts at 0, y is 4)
      * Example
        + System.out.println(“The 1st letter in students name is “ + myString.charAt(0));
        + System.out.println(“The last letter is “ + myString.charAt(myString.length()-1));
        + Potential errors-

“Attempting to access character outside limits of string” means that you are tring to access an index that doesnt exist

**Concatenating Strings**

* The java string class concat() method combines specified string at the end of this string- <https://www.javatpoint.com/java-string-concat>
  + It returns a combined string
* A close-up of a person's face

  Description automatically generated
* Use plus sign to concatenate strings or use concat(stringname)
  + A close-up of a computer code

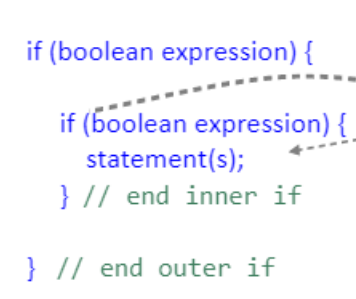
    Description automatically generated
* Converting strings
  + toLowerCase()
    - Returns a string that is in all lowercase
  + toUpperCase()
    - Returns a string that is in all uppercase
* Trim String White Space Characters
  + trim()
  + Returns a string with all white space characters removed from the front or end of the string
  + White space characters include space, tab, line feed, form feed, carriage return
    - Line feeds are \n
  + Ex- String messy = “ long string with white spaces “;
    - System.out.println(messy.trim());
  + Ex with extra formatting-
    - String messy = “ long string with white spaces \n“;
      * System.out.println(messy.trim());
        + Wont print out an extra line

**Flow of control (ifs, if else, nested ifs, multiway ifs)**

* Can either be sequential, conditional (branching/decision), or iterative (looping)
* Relational operators
  + < less than
  + <= less than or equal to
  + > greater than
  + >= greater than or equal to
  + == equal to (NOT ASSIGNED TO)
  + != not equal to
* Logical operators
  + || means or
    - True if at least one of the boolean operands is true, otherwise false
  + && means and
    - True if and only if both operands are true, otherwise false
  + ! means not
    - Reverses the logical value of the expression
* **One way selection structure: if** 
  + ex) if (grade >= 60)
    - {
      * System.out.println(“Passed”);
    - }
* **Two way selection structure: if/else and multiway if**
  + If else example
    - If (grade < 60)
    - {
      * System.out.println(“Come to office hours”);
    - }
    - else
    - {
      * System.out.println(“Passed”);
    - }
  + Complicated example
    - A screenshot of a computer code

      Description automatically generated
  + Multiway if
    - Always ends with else
    - Uses the else if keywords
    - A screenshot of a computer program

      Description automatically generated
* Nested ifs
  + **Use when one condition must be true before checking the next condition**
  + If statements inside of if statements
  + Indents are required
  + Ex- registering for class example
    - Need to check for space in the class first, then check if they have the prereqs to take the class
    - A screenshot of a computer

      Description automatically generated
    - 
* **Concept of scope**
  + When you declare something in an if statement, it is only available to work with within the scope of the if statement
  + If declared outside of the if statement, it can be used in and outside of the if statement
  + A screen shot of a computer program

    Description automatically generated

**Switch and break statements**

* A switch statement lets the value of a variable or expression determine where the program will branch
* Keywords- **switch, case, break, default**
* **DO NOT USE FOR BOOLEAN STATEMENTS**
  + The only time you will use a break is for a switch statement
  + ex) the switch “testExpression” is being evaluated
    - If one of the case “values” matches the value of the switch “expression”
      * The statements for that case are executed
      * Execution stops when a break statement is reached or the end of the switch statement is reached
      * If no case “values” match the values of the switch “expression” a default case is executed if exists
  + Must always be enclosed in parentheses, yield a value of char, byte, short, long, or int type
  + A computer code with text

    Description automatically generated with medium confidence
  + **A screenshot of a computer program

    Description automatically generated**
  + A screen shot of a computer program

    Description automatically generated
* Use an if statement when you need to test for boolean conditions
* Use a switch statement in situations where matching a constant value of type char or int
  + **You cannot use boolean statements in switch and case statements**
  + A pink rectangular sign with black text

    Description automatically generated