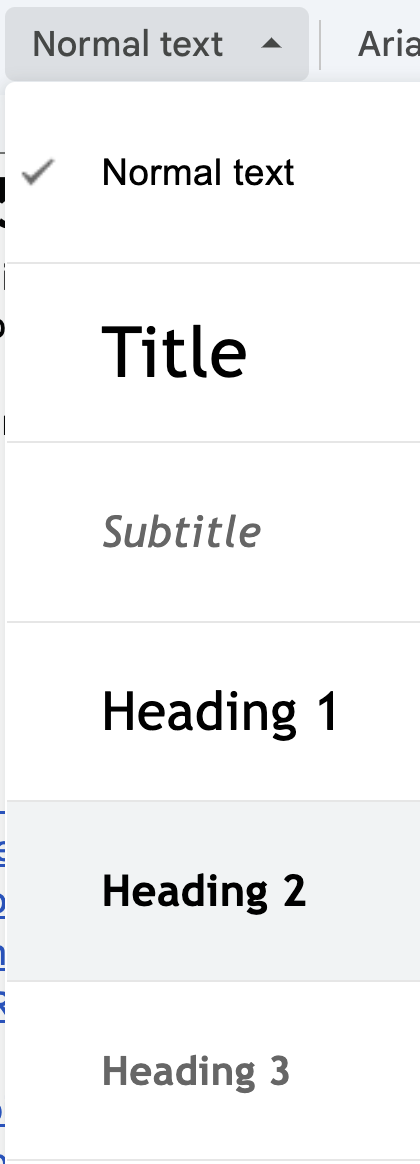
CS1050 Technical Documentation

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

After adding new headings to organize your documentation, refresh the table of contents below.

[Overview of Developer Technical Documentation](#_e7hnk3gmh67o)

[Set Up Development Environment](#_efvt7vjb7dhk)

[General Resources](#_j8bsu2nohj85)

[Module 01: Fundamental Programming Concepts](#_qn332jwlduxf)

[Compilation and Errors](#_8u1ro2aazjxd)

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

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

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

[Code Sample](#_peu7sjhsw2la)

[ANY OTHER TOPIC?](#_9oztm0i5zl6a)

[Module 2](#_qke7paye0rm)

[Module 3](#_345uou64znum)

[Module 4](#_zch7xf7pkk1i)

[Module 5](#_chmkbp3yv5ov)

# Overview of Technical Developmental Documentation

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

Use information from class lectures, guided exploration, code examples from class and book to be able to use later to help you and approach it as if you could help someone else.

GE 01 Put technical documentation below to demonstrate your understanding of Module 1:

* Understand and set up the programming environment (Eclipse, Git, and GitHub).
* Apply version control using Git and document changes effectively.
* Write basic Java programs using variables, constants, and logical structures.
* Debug code using Eclipse IDE and reflect on common syntax and logical errors.
* Develop concise and structured technical documentation.

# Set Up Development Environment

Provide the following but you can copy and paste my steps and add any additional or update my information for you. Make your documentation your own but follow best practices. Create headings that make sense to you.

* **Overview of tools** 
  + Eclipse IDE for Java development
  + Git and GitHub for version control
* **Git and Github**
  + Overview of tools
  + Steps to set up
  + Steps to use desktop ( or command line if you know it) to version locally and back up on server
* **Set Up IDE (Eclipse):**
  + Overview of you IDE
  + Steps to install
  + Configure the workspace to store projects.
  + Verify JDK installation.
  + Creating future projects and class files
  + Use IDE for debugging -

# **Setting up the Development Environment**

***Eclipse:***

* Eclipse is a widely used IDE and is especially exceptional in Java.
* An IDE allows the user to compile, debug, and save code all in one place.
* Eclipse is very easy to use and has features such as correction suggestions.

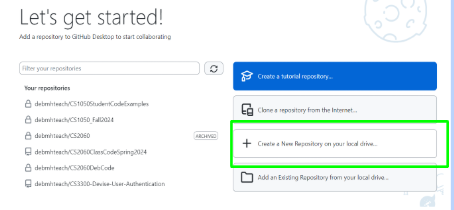
***GitHub & Git:***

* Git: A cloud-based service that allows users to save, backup, compile and upload their code remotely, this is beneficial for version control (updates in code), comment explanation on code, and collaboration remotely with others on coding projects.
* GitHub: This works in unison with Git, it is exactly what it sounds like, it is the hub for all the data stored with Git, it is essentially the UI (user interface making data accessible) of Git, and is where you can upload your data to Git, visualize it, and pull data from Git remotely in order to work further on it.
* Both services work in unison with the IDE.

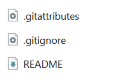
***Git, GitHub, and IDE Eclipse Setup:***

*A free overview of the GitHub and Git -* [*Video*](https://www.youtube.com/watch?v=8Dd7KRpKeaE)

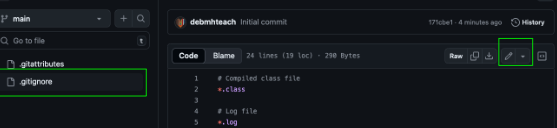
* Download both GitHub and Git
  + [*GitHub Desktop*](https://desktop.github.com/download/)
  + [*Git Download*](https://git-scm.com/downloads)
* Once they have been downloaded, create an account, this will be used to access your projects and remotely access code across all platforms. (It may ask you for an email verification and a few questions, the questions you needn’t worry about)
* Create a folder within GitHub, this is where all your repositories (coding workspaces)
* Once you have completed these steps, you will have the option to create a new repository.



* Once you have filled out your repository to your liking, do not select manage with “external editor” or external IDE.
* Instead, you will select “view files of your repository in the explorer”
* Check to make sure these files are present:



* After this, select “view on GitHub” 
* On the internet version of GitHub, open “.gitignore”.



* Edit this file, and paste the contents of [GitHub ignore](https://github.com/github/gitignore/blob/main/Global/Eclipse.gitignore) into ***YOUR PERSONAL*** GitHub ignore.
* Below the information you just pasted, you must paste the following text:

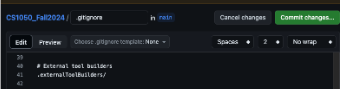
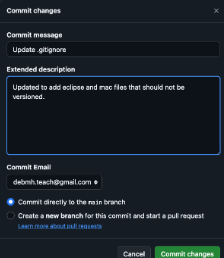
# Eclipse Core

.project

# JDT-specific (Eclipse Java Development Tools)

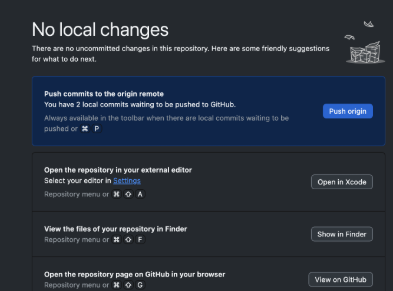
.classpath

* After this is done commit the changes with a comment indicating you have pasted the information.



***Backing Up GitHub Repositories***

* After making changes and committing them, open the desktop app of GitHub
* You should then “push” the committed changes it will look like this:



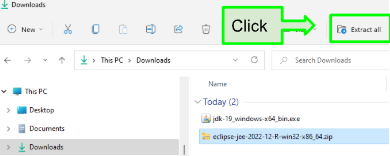
* Selecting “push origin” will back up all changes to the Git cloud service and will update across all platforms.

## Setting up Eclipse (IDE)

* ***First, set-up the JDK (Java Development Kit)***
  + [Download this file](https://drive.google.com/file/d/1vP1itH-tBwCPGa4INFuVmEHXpFdSYnt-/view?usp=drive_link)
  + Once that is done, double click the file, this will start the set-up
  + Allow the app to make changes.
  + Click through the set-up window, default settings will suffice
  + The JDK will end up in program files, this is the correct location
* ***Now set-up the IDE Eclipse***
  + Visit the [IDE website](http://www.eclipse.org/downloads/packages/)
    - This is the Eclipse website
  + Download the Eclipse IDE for Enterprise and Java developers
    - **DO NOT DOWNLOAD INSTALLER**

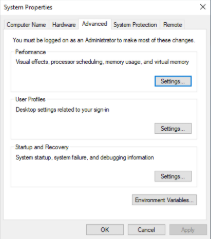
****

* + This is what it will look like.
  + Then you will need to extract the folder, the folder will be in your default downloads folder

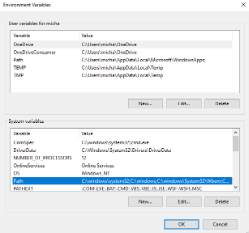


## Setting up environment variables (enabling JDK)

* The file directory of your JDK file after set-up should be **C:\Program Files\Java\jdk-22**
* Now, open settings -> related settings -> advanced system settings



* Once you are there, open “environment variables”.
* Find the path option,

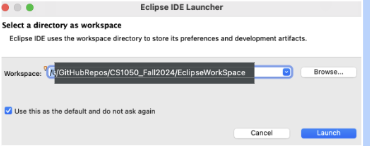


* Now, open edit -> new -> and paste the file directory of JDK 22.

## Setting up Eclipse for use

* Go to the extracted Eclipse file
* There will be a “eclipse.exe” executable file, double click that.
* Find the file directory of the GitHub repository file you made and copy the address into the Eclipse launcher

For example:

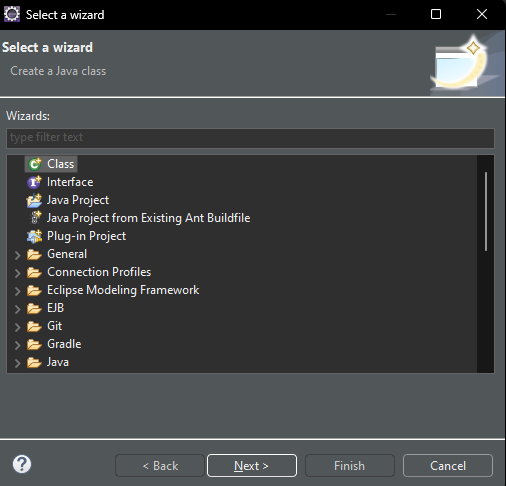


* After that, click launch.
* ***IF YOU MUST UPDATE ECLIPSE CREATE A BACKUP OF YOUR REPOSITORY (GITHUB WORK SPACE)***
* The projects you create in Eclipse will now be stored in your GitHub repository.

## Creating projects in Eclipse

* When creating new classes in Eclipse, you must first create a class file.
  + To create a new class, you must open file -> new -> other -> class

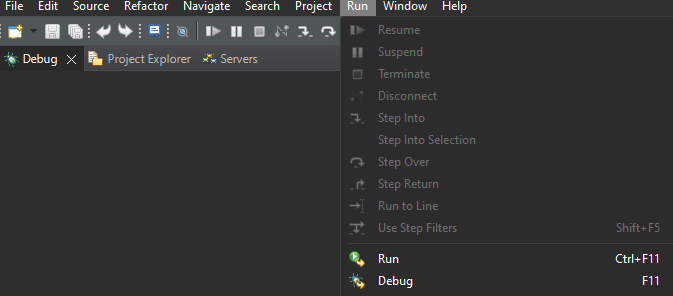
It will look like this



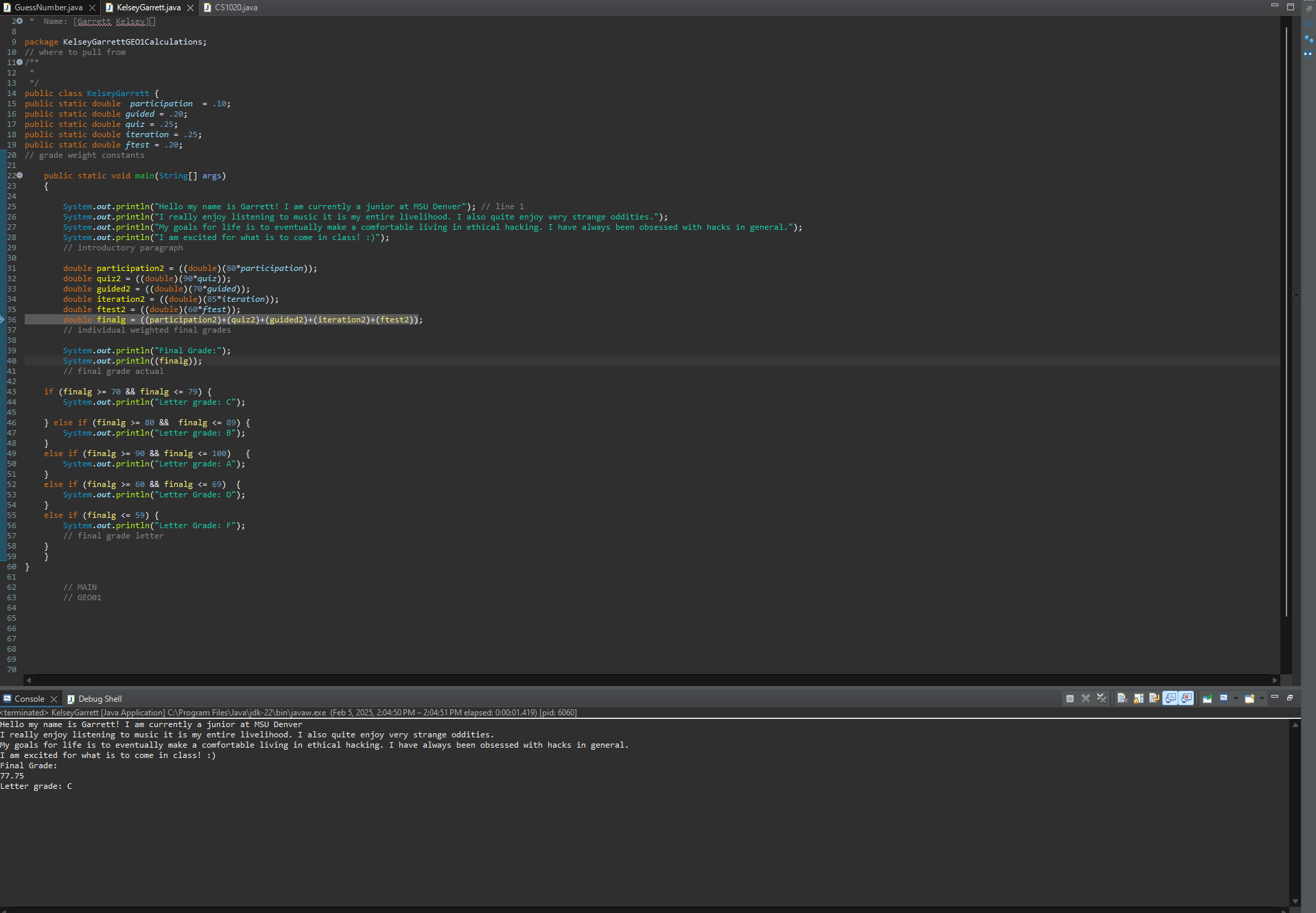
* Once that is completed; by referencing this class, you may start your next project

## Debugging in Eclipse

* Debugging is the process of running your code and “scanning” for errors as a test.
* To access the debugging option go to run -> debug it will look like this:



* Debugging will allow access to the debug window, console, mistake recommendations, and run the code



# General Resources

*Free resources to utilize when looking into the basics of Java.*

* [Shared student CS1050 resources](https://drive.google.com/drive/folders/1HvYY8zzSwlsH--03olvqOJooGnJkZ7F4) – Overall resources in developing your understanding of Java in steps.
* [Java Documentation](https://docs.oracle.com/en/java/) – Overall guide to the building blocks of Java.
* [freeCodeCamp](https://www.freecodecamp.org/news/static-variables-in-java/) – A great resource for tutorials and problem-solving.

## 

# Module 01: Fundamental Programming Concepts

## Common Compilation and Errors

* Syntax Errors – Syntax errors are errors within code that mess with the structure of the code. These errors are caused by things like forgetting to close a block or line of code. For example,  Here, the red underline indicates there should be a “;”. However, because it is not there, the line is not closed and therefore there is a syntax error.
* Runtime Errors – Runtime errors occur when, while running the code, an error occurs. An example of what a runtime error would look like is a math problem that cannot work; therefore, the computer cannot solve it or run it. For example, if there was a division like (1/0) in the code, there would be a runtime error.
* Logic Errors – A logic error is an error that occurs when, despite the syntax of the code being correct, the code is not logically correct. For example, if an integer, a data type of value that cannot have decimals, is used when a decimal is required, there will be a loss of data, and not be correct.

## Primitive Data Types, Variables, Constants, and Naming Conventions

* Boolean data type – A Boolean data type is a single solitary value that represents true or false. Anything that is not 0 is true. So, 1,2,3,… = true 0 = false.
* byte data type – The byte data type is an 8-bit primitive data type; it ranges from -127 to 128. The byte is a small data type, usually used with integers (a keyword used when declaring variables or constants with values). It is in its simplest form, binary code. Byte data, due to its size, is commonly used when conserving memory storage.
* char data type – The char data type is a single 16-bit value, it can be used across a variety of languages and allows Java to initialize software of different languages. It can store any character in a Unicode data set. ([Unicode](https://docs.oracle.com/javase/tutorial/i18n/text/unicode.html) is used when using different national languages as well).
* short data type – Short is a 16-bit data type that, similar to bytes, is a smaller value and can be used when preserving memory storage. However, unlike bytes, the short data type is much more precise.
* int data type – The integer data type represents 32-bit data. An integer is a bigger data type; however, it cannot be used to store decimal data, it can only be used to store whole numbers.
* Long data type – The long data type is a 64-bit data type and has a much bigger range than short, integer, and byte data types. Long data is used when the other data types cannot support a value being used.
* float data type – The float data type is a 32-bit single-precision data type, float data is a larger data type, and has a much higher range of numbers versus a fixed point of data like the others.
* double data type – A double data type is a 64-bit double precision-based data type. It is like a float and is best used when larger decimal numbers are being referenced. However, it is much more precise.

[Java Data Types - javatpoint](https://www.javatpoint.com/java-data-types) – a free resource further describing primitive data types.

[8, 16, 32, and 64 bit data explained](https://howtodoinjava.com/java/basics/difference-between-32-bit-java-vs-64-bit-java/)

**Variables and Naming Conventions**

Store a value that may change (vary), have a data type, stored in computer memory,

have scope - defines

● Naming a variable:

Use meaningful names

Begin with lowercase Multiple-word

identifiers camel case

● Declare and Initialize: can declare

and initialize variables in one step

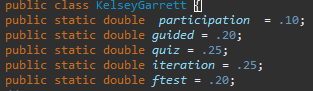
Declare variables by defining what data type (amount of memory needed to store the

value)

Initializing a variable: assign an initial value to a variable (before it is used at all).

### 

**Constants**



Constants are one in the same as variables, declaring a value and preserving appropriate space in the system memory,

* The names should be concise and meaningful
* The data and data types need to make sense
* The declarations should be organized and easy to follow
* Constants are similar to variables, but they cannot change

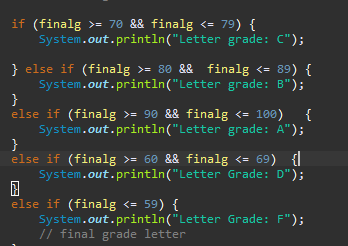
## Arithmetic Operators and Combined Assignment Operators

**Arithmetic Operators:**

* *Symbols that indicate math or arithmetic can be used for things like algorithms.*
* % - modulus division - establishing a remainder value
* + - addition of values
* > , = , < - comparison operators – greater than, less than, equal to, etc.
* \* - multiplication of values
* / - division operator (not when solving for remainder)



An algorithm using the addition of variables.



A series of conditional parameters using comparison operators.

## Memory Allocation, Conversion, and Casting

**Memory Allocation**

The allocation for data is declared by variables or constants while running code. All data types preserve space in the system memory, depending on the data type, it could be a double data type which is a higher bit type and is used to declare larger data types, typically with decimals. On the other hand, the short data type is lesser and typically declares smaller values.

**Casting**

*Datacasting is the transition between data types and the widening or narrowing of the size. There are two types of casting.*

* Implicit casting - Implicit casting refers to a smaller data type to a larger type size. This is like making the byte data type into a double data type. It is essentially widening.
* Explicit casting: Explicit casting is the opposite. It’s essentially narrowing. Casting a bigger type into a smaller type. An example would be an integer, which cannot have a decimal, into a decimal number.

**Conversion**

*The conversion of values, for example, from Int (integer) to Long can be important, for example, in an application that requires large amounts of data that varies, it is safe to assume that the integer may be too small to handle the bigger values so converting integer type to long type is logical.*

Rules in conversion:

Think of conversion in a container scenario. One can convert smaller data types to larger data types (integer to long), but one cannot convert larger data types to smaller ones. (Long to integer). Someone can put a smaller container in a bigger one, however, you cannot put a bigger container in a smaller container. This would result in a runtime error.

## Code Sample

Add code from the guided exploration that contains comments - notes to help you understand the parts of the code.

|  |
| --- |
| The variable grades in each category are multiplied by the category weights.    The constant grade weights in each category.  public static double:  Public – Public constants are declared constants that can be used via multiple class modules, eliminating the need to make a whole new class.  Static - This refers to a type of constant. The declared constant will continue to run through the entire code; there is only one copy of it. Therefore, it can be called upon without the creation of a new class. - [free resource to codeCamp's guide to static variables/constants](https://www.freecodecamp.org/news/static-variables-in-java/)  Double – The keyword double is a data type that allocates storage in system memory. In this case, double is used because the data or values are in decimal form. However, in the case of the int finalg it is converted to a whole number as final grades are typically rounded, so it is easier to dictate a final letter grade. |

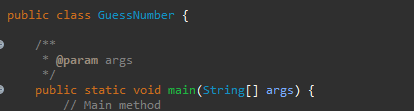
## ANY OTHER TOPIC?

# Module 2: Predefined Classes, Methods, and Decision Structures

## Classes, Objects, and Methods

*Classes:*

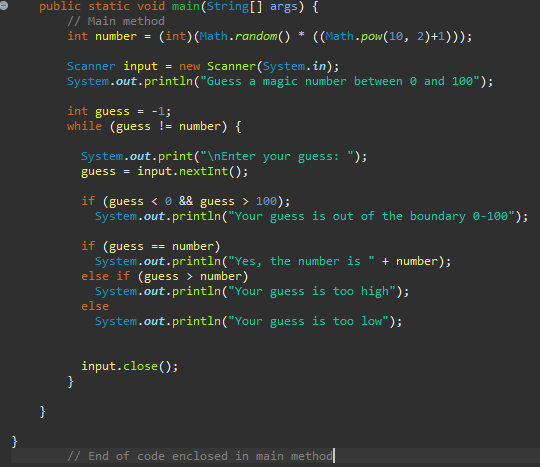
A class is essentially a blueprint, it encapsulates your method, objects, and other attributes in your code. These are typically spelled with a beginning capital letter. At least one class is needed in every Java program.



- The class is created, “GuessNumber”, then the method is called, and the rest of the code follows, all within the class.

*Methods:*

A method is what encapsulates a block of code, it instructs the computer what code to run. It is within the class and is declared at the beginning.



* The calling of the method is not identified, all the code within the method’s block is indented
* The bracket that closes out the method is always the last one

*Method Signature, Parameters, and Return Values:*

*Method Signature –*

*Predefined Classes:*

Predefined classes are default classes within Java that do not need to be imported or created.

* Examples:
  + The Math class – This class is a predefined class within Java itself, it does not need to be imported or called upon and is used for things like mathematic equations, random numbers, exponential numbers, etc.
  + The System class – This class is a predefined class within Java itself as well, it need not be imported and contains things like input streams and output streams, this can be used to print text.

*Non-predefined Classes:*

Non-predefined classes are classes that need to be created or imported.

* Examples:
  + The Scanner class – The scanner class is apart of the *java.util* package. Thus, the java.util package must be imported in order to utilize this class. It is usually used for things like keyboard inputs.

 - Importing the java.util package in order to use the scanner class, this is done at the very beginning of your code. - Utilizing the scanner class to create a new object to record keyboard input.

# Module 4

[ADD Headings 2 for the different topics then add your documentation from the lectures]

# Module 5

[ADD Headings 2 for the different topics then add your documentation from the lectures]