**CS1050 Technical Documentation**

[**Overview of Developer Technical Documentation**](#_Toc184299140)

[**Set Up Development Environment**](#_Toc184299141)

[Version Control with Git and GitHub](#_Toc184299142)

[IDE Information](#_Toc184299143)

[**General Resources**](#_Toc184299144)

[**Module 1**](#_Toc184299145)

[Compilation process](#_Toc184299146)

[Variables, Constants, and Primitive Data Types](#_Toc184299147)

[Arithmetic Operators and Combined Assignment Operators](#_Toc184299148)

[Memory Allocation, Primitive Data Types, Conversion and Casting](#_Toc184299149)

[Quality Code: Naming Conventions](#_Toc184299150)

[**Module 2**](#_Toc184299151)

[Classes, Objects, and Methods](#_Toc184299152)

[Conditions](#_Toc184299153)

[Methods](#_Toc184299154)

[Scanner and Math Class](#_Toc184299155)

[String and char](#_Toc184299156)

[Nested if, Multi-way if, and Switch](#_Toc184299157)

[Character Class](#_Toc184299158)

[**Module 3**](#_Toc184299159)

[Loops](#_Toc184299160)

[Nested Loops](#_Toc184299161)

[Methods](#_Toc184299162)

[Stack frame](#_Toc184299163)

[**Module 4**](#_Toc184299164)

[Arrays](#_Toc184299165)

[1D arrays](#_Toc184299166)

[2D Arrays](#_Toc184299167)

[1D Array (Using a List)](#_Toc184299168)

[2D Array (Using a List)](#_Toc184299169)

[Examine an Array of Strings](#_Toc184299170)

[Rules for arrays](#_Toc184299171)

[Stack and heap memory](#_Toc184299172)

[Passing arrays and reference variables to methods](#_Toc184299173)

[Returning arrays from methods](#_Toc184299174)

[Sorting and searching arrays](#_Toc184299175)

[**Module 5**](#_Toc184299176)

[Classes and objects](#_Toc184299177)

[Constructors](#_Toc184299178)

[Static variables and methods](#_Toc184299179)

[Encapsulation](#_Toc184299180)

[Association](#_Toc184299181)

[Aggregation and Composition](#_Toc184299182)

[Inheritance](#_Toc184299183)

[Exception Handling](#_Toc184299184)

[Reading from a File](#_Toc184299185)

[Writing to Files](#_Toc184299186)

[Terminal Commands and Git/Github](#_Toc184299187)

# **Overview of Developer Technical Documentation**

My technical documentation includes:

* **Setup instructions**: Steps to configure my development environment (GitHub)
* **Version control**: Key commands for Git and how they work
* **IDE information**: Setting up and using IntelliJ IDEA, including downloading, installing JDK, and creating a new project.
* **General Resources**: Links to important resources
* **Modules**: Topics covered in modules

A screenshot of a computer

Description automatically generated

# **Set Up Development Environment**

1. Login to [GitHub](https://github.com)
2. Click on the “+’ icon and select “New repository.”

A screenshot of a computer

Description automatically generated

1. Enter a name and add .gitignore file then create the repository.
2. Go to IntellJ IDEA > File > Settings > Version Control > Git
3. Make sure that the path to Git is set correctly.

## 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 is a distributed version control system that tracks changes in code over time. It allows multiple people to work on the same project without messing up each other’s work.

GitHub is a developer platform that allows developers to create, store, manage, and share their code. It uses Git software to track changes and manage versions of code.

Important GitHub commands:

**git init** – Makes a repository (Only done once per repository)

**git status** – Extremely useful to check now and then

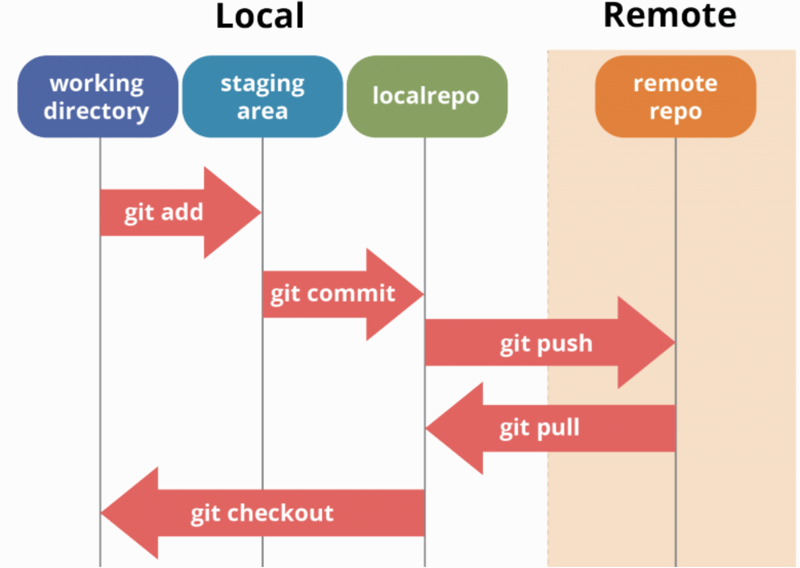
**git add** – Start tracking or include changes for a file

**git commit -m “message”** – Commit the changes with a message

**git push** – Push the branch to GitHub (remote)  
**git pull** – Pull the changes for the current branch

**git checkout -b** – Make a separate branch

**git reset - -hard** – Resets back to commit   
**git branch -D** – Deletes the branch



## IDE Information

Setting up a project with IDE

1. [Download JetBrains IntellJ IDEA](https://www.jetbrains.com/idea/download/?section=windows)  
   Choose windows  
   Complete Setup, and install required plugins
2. [Install JDK](https://www.oracle.com/java/technologies/downloads/)  
   JDK is required to compile and run Java applications
3. Create a New Project from the welcome screen  
   Click **File, New, and Project** if the welcome screen isn’t displayedA screenshot of a computer

   Description automatically generated
4. Right-click on your project and create a Java Class  
     
   A screenshot of a computer

   Description automatically generated
5. [More information (includes running, registering, updating, uninstalling, creating Java applications, features, plugins, etc.)](https://www.jetbrains.com/help/idea/run-for-the-first-time.html)

# **General Resources**

Here you can list resources that you use frequently.

* [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)
* [Syllabus](https://msudenver.instructure.com/courses/95281/assignments/syllabus)
* [Announcements](https://msudenver.instructure.com/courses/95281/announcements)
* [Modules](https://msudenver.instructure.com/courses/95281/modules)

# **Module 1**

## Compilation process

You write code in a “.java” file. The Java compiler converts it to bytecode in a “.class” file. The JVM (Java Virtual Machine) then executes this bytecode, making the program runnable. (IDEs do this for you

## Variables, Constants, and Primitive Data Types

Variables are containers for storing data values.  
(String, int, float, char, boolean)  
  
If you don’t want others to overwrite existing values, use the “final” keyword. This will make it a constant. (Unchangeable)

This website describes and has examples of variables and constants.  
<https://www.w3schools.com/java/java_variables.asp>  
  
This website describes and has examples of primitive data types.  
<https://www.w3schools.com/java/java_data_types.asp>

## Arithmetic Operators and Combined Assignment Operators

Arithmetic Operators in Java perform basic mathematical operations.

A table of maths

Description automatically generated with medium confidence

Combined Assignment Operators make it easier to update variables.  
  
Example:  
a += 3 instead of a = a + 3  
b \*= 2 instead of b = b\* 2

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

Memory allocation is reserving space in a computer’s memory for storing data.  
Declaring a constant allocates enough memory for the data type in read-only memory.

Declaring variable allocates enough memory for the data type in read/write memory.

Conversion is changing data from one type to another.   
Example: Converting string to integer

Casting is converting one data type to another.

1. Widening (implicit casting):

* Casting a value with a smaller range to one with a bigger range
* Java does this automatically
* ex – double d = 3;

1. Narrowing (explicit casting):

* Casting a value with a bigger range to one with a smaller range
* You must do this explicitly because it can result in losing data
* ex – int number = (int)3.0;

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

* boolean data type
* byte data type
* char data type
* short data type
* int data type
* long data type
* float data type
* double data type

## Quality Code: Naming Conventions

Naming conventions are important for maintaining readable and clean code.

1. Constants:  
   - Use UPPER\_CASE with underscores separating words.  
   - Example: “TECHINCAL\_DOCUMENT”, “FINAL\_GRADE”
2. Variables:

* Use camelCase with the first letter lowercase.
* Example: “techincalDocument”, “finalGrade”

A screen shot of a computer

Description automatically generated

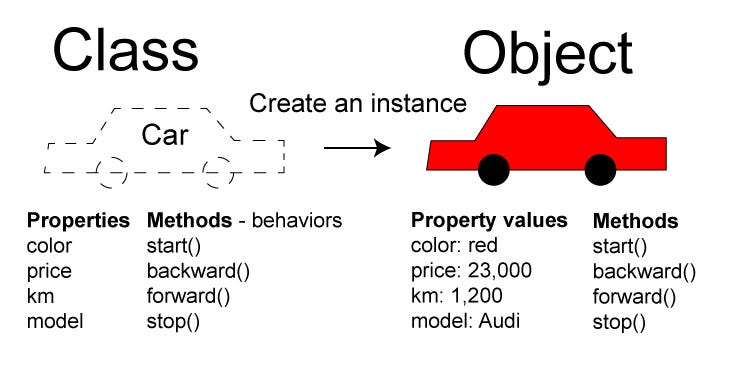
# **Module 2**

## Classes, Objects, and Methods

A class is a blueprint used to construct an object. It defines an object’s attributes and behaviors.

Example: Car has color, model, and price.

A class is not an object. An object is created using the class blueprint.  
Example: You can make an actual car like a red or a blue car.  
You can create many objects from the same class.

A method is like a function or action that objects of the class can perform.  
Example: Car class methods could be “start”, “forward”, “stop”  
  


## Conditions

Condition is an expression that evaluates to ‘True’ or ‘False’. Conditions are used to make decisions.

Example: if (age > 21), this evaluates to True if age is greater than 21, otherwise it’s False.

* == (equals to)
* != (not equal to)
* > (greater than)
* < (less than)
* >= (greater than or equal to)
* <= (less than or equal to)
* || (or)
* && (and)
* ! (not)

## Methods

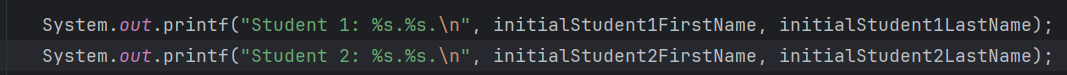
Methods are used to have a computer program do an action.



(Call method print and pass argument string literal)

Formatting data with Printf

* %b (boolean value) (true or false)
* %c (character) (‘a’)
* %d (decimal integer) (200)
* %f (floating-point number) (45.46000)
* %e (number in standard scientific notation) (4.556000e+01)
* $s (string) (“Java is cool”)

Example usage: 

(%s. %s. inserts value from initialStudentFirst, initialStudentLast separated by a dot)

## Scanner and Math Class

Scanner class: First, import the scanner class, then create a scanner object, then use scanner methods.

A computer screen shot of a program code

Description automatically generated

Math class: Doesn’t need an object like for Scanner class. Has methods that allow you to perform mathematical tasks on numbers.

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

This website has a list of math methods for performing basic numeric operations like logarithm, cube root, and trigonometric functions.

## String and char

A string is in double quotes, while a char is in single quotes.

A string variable can hold multiple characters, and a char variable can hold only one.

A screen shot of a computer program

Description automatically generated

Note: input.next() reads until whitespace. Input.nextLine(): reads the whole line.

**String Methods:**length() = Returns the number of characters in a string.

charAt(index) = Returns the character at the specified index within a string.

Example: Char is the first letter at index 0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| H | E | L | L | O |
| 0 | 1 | 2 | 3 | 4 |

## Nested if, Multi-way if, and Switch

Nested if: One condition must be true before checking the next condition.

A screen shot of a computer program

Description automatically generated

Multiway if: Only one condition can be true at a time.

A computer screen shot of text

Description automatically generated

Switch: Dealing with multiple specific values of a variable.  
A computer screen with text on it

Description automatically generated

## Character Class

Character class contains methods that do tests on characters. Doesn’t need an object to be created for this class.

Predefined Methods of Character Class:

* Character.isDigit() = True if the character value is a digit
* Character.isLetter() = True if the character value is a letter
* Character.isLetterOrDigit() = True if the character value is a letter or digit
* Character.isUpperCase() = True if the character value is in uppercase
* Character.isLowerCase() = True if the character value is in lowercase
* Character.toUpperCase () = Converts to uppercase letter
* Character.toLowerCase () = Converts to lowercase letter

# **Module 3**

## Loops

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **For Loop** | **While Loop** | **Do While Loop** |
| Syntax | for (initialization; condition; increment) | while (condition) | do { … } while (condition) |
| Execution | Executes a block of code for a specific number of iterations | Executes a block of code as long as the condition is true | Executes a block of code at least once, then checks the condition |
| Initialization | Done in the loop statement | Must be done before the loop | Must be done before the loop |
| Condition | Checked before each iteration | Checked before each iteration | Checked after each iteration |
| Use Case | Iterating a known number of times | Iterating an unknown number of times until the condition is met | Ideal when the block of code needs to execute at least once |
| Example | for (int I = 0; I < 5; i++ { … } | while (I < 5) { … } | do { … }  while (I < 5); |

## Nested Loops

A screen shot of a computer

Description automatically generated

Nested loops are loops inside other loops. The inner loop runs completely for every iteration of the outer loop.

* The outer loop iterates with outerNum, it takes values from 1 to 3 and prints each value of outerNum.
* The inner loop iterates with innerNum taking values from 1 to 2 and prints each value of innerNum.
* The inner loop executes completely for each iteration of the outer loop.

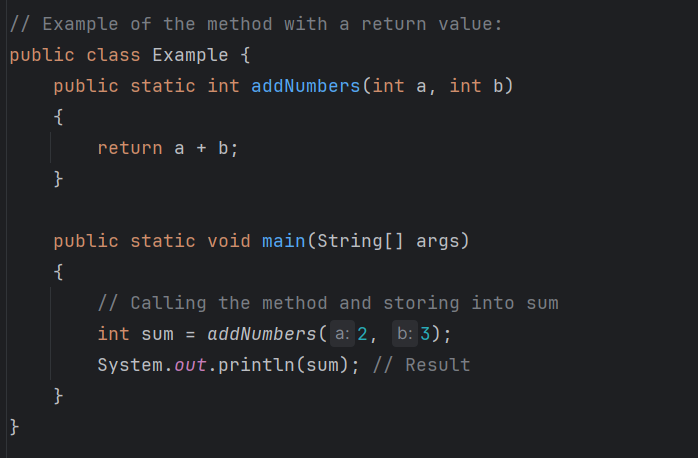
## Methods

Methods are blocks of code that perform specific tasks and can be called multiple times within a program. Each method can take parameters and return a value.  
Methods can either return a value or be void. Void means it doesn’t return anything.

Example of the method without a return value:A screen shot of a computer program

Description automatically generated

Example of the method with a return value:



## Stack frame

A stack frame is a data structure that holds information about a method’s execution in a program. When a method is called, a new stack frame is created and pushed onto the call stack, and when the method returns, the frame is popped off.

Push: Adds an element to the top of the stack. This increases the stack’s size by one.  
Pop: Removes the top element from the stack. This decreases the stack’s size by one and returns the removed element.

A diagram of a bar chart

Description automatically generated with medium confidence

Website for Stacks, it’s in C++ but same concept:  
<https://www.w3schools.com/cpp/cpp_stacks.asp>

# **Module 4**

## Arrays

An array is a data structure that provides a way to store more than one value. Arrays must be the same data type.   
  
There are several types of arrays:

## 1D arrays

This is how you declare and initialize a 1D array:   
A screen shot of a computer code

Description automatically generated

## 2D Arrays

A 2D array is an array of arrays, that allows you to store data in a grid-like format (rows and columns).

Declaring and initialize a 2D array:  
A computer screen with text

Description automatically generated

## 1D Array (Using a List)

List<Int> oneDArray = Arrays.asList(1,2,3,4,5)

## 2D Array (Using a List)

List<List<Int>> twoDArray = Arrays.asList(

Arrays.asList(1,2,3),  
 Arrays.asList(4,5,6),

Arrays.asList(7,8,9)

);  
  
More information: <https://www.w3schools.com/java/java_arraylist.asp>

## Examine an Array of Strings

When you create an array of strings, here’s what hapenes:

String[] stringArray = (“Apple”, “Banana”, “Cherry”};

1. Stack Memory: The stringArray variable is stored on the stack.  
   Each element in the array is a reference pointing to the actual String objects.
2. Heap Memory: The String objects are stored on the heap.  
   The array itself is also stored on the heap.

## Rules for arrays

1. When an array is created, its size is fixed. You can’t change the size of an array after it’s initialized.
2. The index for arrays starts at 0. The first element is at index 0, then 1..2..3.. etc.
3. You can get the length of an array using the .length attribute.
4. Everything inside the array needs to be the same type. For example, you can’t mix strings and integers. It can’t hold elements of mixed types.
5. If an array is created but not initialized, it will default to the value 0 for integers and null for reference types.
6. ArrayIndexOutOfBoundsException happens when the index is outside the bounds of an array (Remember the array starts at index 0).

## Stack and heap memory

Declaring an array creates memory on the stack for the numbers variables to store the address to refer to where the array will be in the memory on the heap.

Stack holds primitive data types like int, double, char, etc. Each variable holds its actual value directly in memory. Every time a method is called, a new stack frame is created. When the method is completed, the stack frame is destroyed.

Array names contain memory address on the heap where array starts at index 0. The heal is not on the stack.

For example, in int []numbers = new int[5]  
The array object is stored in the heap

## Passing arrays and reference variables to methods

When an array is passed to a method, it’s passing the reference to the array, not a copy of the array itself. This means that any changes made to the array elements inside the method will affect the original array outside the method.

Reference variables are passed by reference, which means the method can change the object that the reference variable points to.

## Returning arrays from methods

When returning an array from a method, it’s returning the reference to the array, not a copy.

Returning reference variables is similar to returning arrays, it returns the reference and not a copy. Changes to the returned object will affect the original object.

An array of strings stores references to String objects.

So in a stack, the array variable names stores a reference to the array of strings.  
Each element (car[0], car[1], etc.) holds a reference to a String object.

The array object itself is stored in the heap.  
The string objects (“Ford”, “Mustang”, etc.) are stored in the heap.

## Sorting and searching arrays

Ways to Sort: <https://www.thedshandbook.com/sorting-in-arrays/>

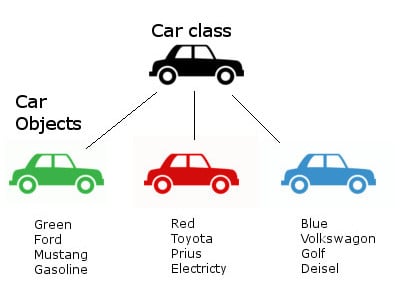
Ways to Search: <https://www.thedshandbook.com/searching-in-arrays/>

# **Module 5**

## Classes and objects

A class is a blueprint for creating objects. It defines the properties and actions that the objects will have.

Example:



Attributes: These are variables that hold data about an object.

Example: In a car class, attributes are color, make, models, etc.

Methods: These are actions that objects can do.   
Example: In a car class, methods could be drive() or brake().  
  
More information: <https://www.w3schools.com/java/java_oop.asp>

## Constructors

A constructor has the same name as the class.

A constructor doesn’t have a return type and doesn’t need to include void.

It’s optional for your classes.

A screenshot of a computer code

Description automatically generated

## Static variables and methods

Static variables: A variable that is attached to a class rather than to an instance of the class.

These are shared by all instances of the class.

Static methods: A method that is attached to a class rather than an instance of a class.

Static methods can only access static variable directly.

## Encapsulation

Encapsulation is to make sure that “sensitive” data is hidden from users.

How to:

Declare class variables/attributes as private.

Provide getter and setter methods to access and update the value of a private variable.

Getter and setter methods:  
A screen shot of a computer code

Description automatically generated

More information: <https://www.w3schools.com/java/java_encapsulation.asp>

## Association

A relationship that describes an activity between two classes where there is a loose connection, they exist independently.

For example, a student class can borrow books from a library, but a student's existence doesn’t depend on the library, and vice versa. They are loosely connected through the borrowing activity.

## Aggregation and Composition

Aggregation is a special form of association, that represents an ownership relationship between two objects.

For example, a department has professors, but professors can exist independently of a department.

## Inheritance

**Inheritance** is a mechanism where one subclass acquires the properties and behavior of another superclass, enabling code reuse and hierarchy.

**Superclass**: The parent class that provides common attributes and methods.

**Subclass**: The child class that inherits from the superclass and can add or override features.

**Method overriding:** When a subclass modifies the implementation of a method defined in its superclass.

Example:

class Animal { // This is a superclass

void eat() {

System.out.println(“This animal eats food.”);

}

}

Class Dog extends Animal { // This is a subclass

Void bark() {

System.out.println(“Woof”);

}

}

Example from code:  
A computer screen shot of a black background

Description automatically generated

## Exception Handling

1. Try Block: Code that might throw an exception.
2. Catch Block: Handles the exception if it occurs.
3. Finally Block: Executes code regardless of whether an exception was thrown or not.
4. Throw: Used to explicitly throw an exception.
5. Throws: Declares exceptions a method might throw.

Java provides a number of predefined exception classes:

* Error
* Exception
* RuntimeException
* ClassNotFoundException
* NullPointException
* ArithmeticException
* Etc..

A screen shot of a computer program

Description automatically generated

## Reading from a File

When the file is opened, a file marker will be placed at the first character in the file.

When the following line of code executes, the next() method reads a string up to but not including the delimiter.

String type = inputFile.next();

When the follow line of code executes, the next() method reads a string up to the delimiter.

String name = inputFile.next();

There is a trim() method on strings that removes all leading or trailing spaces.

Then continue using the Scanner methods to read the information on the line.

Example:

A computer screen with white and orange text

Description automatically generated

This is reading the instructor’s name and email from a file and setting it.  
  
More information: <https://www.w3schools.com/java/java_files_read.asp>

## Writing to Files

You can open, read, and write data to files from a program. A file path is the “address” of a file on the computer, showing where it’s stored. You can write to a file by creating a PrinterWriter object.

Example:  
A computer screen shot of a program code

Description automatically generated

More information: <https://www.w3schools.com/java/java_files_create.asp>

## Terminal Commands and Git/Github

For Git/GitHub information and command line, go to [**Set Up Development Environment**](#_Toc181886210)

T**erminal commands**

ls – Lists files and directories.

cd [directory] – Changes the current directory

For more commands: <https://devblogs.microsoft.com/scripting/table-of-basic-powershell-commands/>

(Note to future self)

Completion of CS 1050: This document serves as a foundation. For advanced topics, refer to the technical document created in CS 2050.