CS 1050 Technical Documentation

General Resources

Your first source should always be:

- Your class lecture slides, documents and the resources linked in the canvas materials, instructor (Deb), learning assistant (LA)

If you need extra clarification or examples you can use these but always compare with the lecture information.

- Runestone Academy – CS Java (interactive book): https://runestone.academy/ns/books/published/csjava/index.html
- GeeksforGeeks Tutorial– Programming Concepts & Java: https://www.geeksforgeeks.org/java/
- W3Schools – Java, syntax, data types, etc.: https://www.w3schools.com/java/
- Oracle Official Docs – For advanced lookups: https://docs.oracle.com/javase/tutorial/java/index.html

If you're unsure whether a source is trustworthy, ask your instructor first.

⚠️ Avoid These Sources

- TPointTech https://www.tpointtech.com/
- Sites without a known author or date
- AI-generated blogs that do not cite documentation

Tools:

- Visualize Java Code - used to step through code and edit code
-

Feel free to add more resources shared from the lectures.

Development Environment

Summarize version control and Integrated Development Environment .Include an images you think is helpful and include the link to the website containing the image

Git and GitHub

Summarize git and github.

- Difference between git and github
- Local git repo with Screenshot of your local git repository folders and files
- github desktop version control steps with commands and their description
- Github remote repository with screenshot of Eclipse Project Folder and java files
- Gitignore file
- Resource List to help you

Eclipse IDE

Summarize Eclipse and include resources that will help you use eclipse.

Module 1: Language Fundamentals

Here are examples of technical documentation to help you get started.
Notice how I didn't copy and paste from the lecture.

1. I create a heading 2 to make a section for different parts of a java program
2. I used code from the lecture to explain in my own words after reading M01 L02 slides 6 - 17 to summarize the concepts
3. I put code in a table and below the table I explained different parts

Feel free to update the following to help you understand.

Parts of a Java Program

```
public class ComputeArea {
  public static void main(String[] args) {
       //declare and initialize constant in read only memory
       final double PI = 3.14159;
       //declare variables will allocate memory based on data type.
       double radius;
       double area;

       radius = 20;

       double area = radius * radius * PI;

       System.out.println("The area for the circle of radius " +
    radius + " is " + area);
  }//end of main
}//end of class
```

- Keywords are defined already for java: **public ,class, static, void, final, double**
- Identifiers are names made by the programmer for classes, methods, variables and constants
    - Camel case is how you write the names with multiple words
    - Use meaningful names
    - Use correct naming conventions Naming Conventions - Java
        - Classes start with capital letter: ComputeArea
        - Variables start with lowercase letter: radius
        - Constants all uppercase with _ to separate words: PI
- Every program has a class :
    - Has { where class block starts and } where it ends
    - Example: ComputeArea
- Inside the class is the main(String[] args) method
    - Has { where main method block starts and } where it ends
    - Method block contains code statements that will run
- Print methods
    - It prints what is in the parenthesis in quotes “” and in the variables
    - + is used
        - Example: System.out.println("The area for the circle of radius " + radius + " is " + area);
- A literal is a fixed value used in code
    - example 3.14159, 20
- Variables store different types of data in memory

    - The data type: determines how many bytes to store that type of data such as double or integer which are
    - primitive data types
    - Declare variables with data type identifier to allocate memory
        - Example: **double** radius;
    - Initialize a variable means to assign the first value to store in memory
        - Example radius = 20;
    - The = means assign what is on the right and store in memory identified by name on the left
    - You can declare and initialize in one step
        - Example: **double** radius = 20;
    - Always initialize variables before using
- Constants store values in read only memory because the don't change:
    - When declaring a constant use the final keyword then the data type before assigning
    - Example: **final double** PI = 3.14159;
    - Use all caps identifier for constant
- Comments are used to explain code
    - Ways to write Java Comments
    - Example: //end of main

Resources:

- M01 L02 Language Fundamentals
- 1.3. Variables and Data Types — CS Java

Operations and Evaluating Expressions

Order of Operations:

- Parenthesis
- Exponents
- Multiplication / Division (Left to right)
- Addition / Subtraction (Left to right)

Some other operators used in coding are:
% = Mod operator shows remainder after division (20 % 6 = 2 because 6 goes into 20 3 times with 2 left over)
- is used to determine even or odd numbers in coding (variable % 2 == 0 is even, variable % 2 == 1 is odd)
|| = or
&& = and
!= = not equals
"==" = literal equals
"=" = assignment operator, not "equals"
< = less than
<= = less than OR equal to
> = greater than
>= greater than OR equal to

```java
if(calcFunction.equals("add") || calcFunction.equals("+") ) {
    double solution = firstNumber + secondNumber;
    System.out.println(firstNumber + " + " + secondNumber + " = " + solution);
}
else if(calcFunction.equals("subtract") || calcFunction.equals("-") ) {
    double solution = firstNumber - secondNumber;
    System.out.println(firstNumber + " - " + secondNumber + " = " + solution);
}
else if(calcFunction.equals("multiply") || calcFunction.equals("*") ) {
    double solution = firstNumber * secondNumber;
    System.out.println(firstNumber + " * " + secondNumber + " = " + solution);
}
else if(calcFunction.equals("divide") || calcFunction.equals("/") ) {
    double solution = firstNumber / secondNumber;
    System.out.println(firstNumber + " / " + secondNumber + " = " + solution);
}
```

```java
static void intAssignment() {
    int myAge = 12;
    System.out.println("myAge is " + myAge);
    System.out.println(myAge = 99); // x is changed and the value of that assignment printed
    System.out.println("myAge is now " + myAge);
}

static void booleanAssignment() {
    boolean value = false;
    System.out.println("value is " + value);
    System.out.println(value = true);
    System.out.println("value is now " + value);

}
```

Organize your documentation for L03 below. Create headings and add explanations with code examples.

Version Control and IDE

Git provides version control for your files
Allows for you to roll back files to older versions if something breaks
Work in branches

GitHub provides a cloud based service to store your local Git versions of files;
Can collaborate with pull/push/merge while avoiding issues or reverting if breaks
Repositories(local and remote) are locations where versions are stored
Always provide detailed commit messages

Primitive Data Types and Casting

Data types can be implicitly cast to a larger data type(ie, int to double, int to long, etc)
Must be explicitly cast to a smaller data type (double to int)

Type	Size (in bits)	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2^{31} to $2^{31}-1$
long	64	-2^{63} to $2^{63}-1$
float	32	1.4e-045 to 3.4e+038
double	64	4.9e-324 to 1.8e+308
char	16	0 to 65,535
boolean	1	true or false

Startertutorials.com