CS 202-1 Chapter 1

August 29, 2018 2:34 PM

- A **program** is simply a set of instructions for a computer to follow
- You can **run** or **execute** a program on the data
- A program uses data as its **input**
- The program will **output** a result based on the data that put in
- The computer and the program are considered to be one unit
- The first program that you use and run on the computer is the **operating system**
- **Software** is another word for programs
- **Java** is a very easy to use language compared to other programming language
- Java is a **high-level language**
- **Assembly language** is machine language but easier to read for people
- After some additional translation, Assembly language becomes **low-level language**
- The language a computer directly understands is called **machine language** or binary code
- For some high-level languages you can use a **compiler** to translate or **compile** the language
- A compiler can create an **executable** program that can run as many times as you want
- An **interpreter** is a program that alternates the translation and execution of statements in a program written in high-level language
- The **Java compiler** translates your Java program into a language called **bytecode**
- Bytecode is a machine language for a hypothetical computer known as a **virtual machine**
- **Java Virtual Machine,** or **JVM** is a kind of interpreter that translates bytecode into a read-able language or machine language for an actual computer
- The **run command** tells the bytecode interpreter to execute the bytecode
- Most Java programs consist of many different pieces of code or parts called **classes**
- In order to connect these pieces or classes we use a program call a **class loader**
- The **syntax** of a programming language is the set of grammatical rules for the language or it's the grammar of the language
- A **syntax error** violates the grammar or rules for a language
- A **logic error** occurs when a solution to a problem is not correct
- A **run-time error** occurs while running the program and will often abruptly halt the program by manifesting itself; usually a file is missing or the name of the file being inconsistent
- **Bytecode** is a very portable coding language because bytecode can be transferred across the internet and run on any type of computer operating system
- Compiler to bytecode to machine language
- **Source code** is a text based code written by the programmer and in Java it will end with **.java**
- Bytecode is a converted source code that the ends with **.class**
- **Applications** run on a specific computer
- **Applets** run through a web browser
- It's important to understand the problem in your programs, plan your logic effectively, code the program, compile the program, test the program, and implement the program
- The file name of the program needs to **match** the class name in the program exactly
- **javac FirstProgram.java** is used to compile the program in Java
- **Debugging** is the process of fixing the bugs in your program
- In order for the **changes** that you make to take effect you need to **save your work** and **re-compile** the program.
- **Running** or **executing** is done by using **java FirstProgram** not **javac FirstProgram.java**
- **Testing** involves eliminating those logic errors or fixing those syntax errors, and sometimes fixing run-time errors
- The braces in the code are call **methods** and every java application has a method called **main**
- A **statement** is an instruction within a method to define a task and make up the **body** of the method
- Java programs use **software objects** or **objects** to perform actions
- The items inside parentheses are called **arguments** and provide the information the method needs to carry out its action
- A Java program has two files: source code file, and a class file

General

- Put your variables at the very top of your program just after the main method
- An object, in a Java program, performs and action when you **invoke**, or **call**, on of its methods
- All code written inside of the braces will be indented 3 spaces

Algorithms

- An **Algorithm** is a set of directions for solving a problem

Planning

- **Pseudocode** - is a quick way of describing a solution and is regular English
- **Flowchart -** is a graphical or visual representation of logical steps for a program
- Flowchart symbols and shapes: Input/output - slanted rectangle, processing - rectangle, start/stop - horizontal bubble, connectors - small circle/square with triangular bottom, declaration - slanted horizontal bubble

CS 202-1 Chapter 2

September 10, 2018 8:06 AM

Variables

- Variables store data to be used in a program
- Variables also define the type of operation being used
- Variables must be declared before they are used in a program
- All variables need a proper name
- Values are the data that a variable stores for later use in a program
- Only one value can be used in one variable
- Remember to select meaningful and proper names for variables
- Variable names are also called identifiers
- Variables are stored in RAM as an address
- Identifier names can use letters, digits, and the underscore character: _ (you cannot start the name with a digit and you cannot have spaces between words)
- For this class we will use camel-casing (separating words with capitalization); example: numOfStudents, not num_of_students
- Data types - specifies a set of values, and the operations that can be performed with those values
- Java has 8 primitive data types: (byte, short, int, long, float, double, char, boolean)

Data Types

- Data types - specifies a set of values, and the operations that can be performed with those values
- Java has 8 primitive data types: (byte, short, int, long, float, double, char, boolean)
- Every variable in a Java program must be declared before it can be used in a program
- There are 4 different parts to each variable: **(name, type, value, and address)**
- The name of a variable is directly connected to the address to where it's located in RAM
- Primary integer types: (byte, short, int, long)
- Floating-point: (float, double) - remember to add a lower case "f" to the end of each floating-point number
- Character: (char) - use single quotes around the characters
- Boolean: (boolean) - can only be true or false
- String: (String) - a string of 0 or more characters enclosed in double quotes ("") - not a primitive data type

Declaration

- Every variable must be declared before it is used for the very first time
- All variables must first appear on the left side of the assignment operator
- Assignment Examples:
  - type name;
  - variable = expression;
  - The operand '=' is called "gets"
  - The variable name must be on the left side of the operator
  - area = radius * radius * PI
- Combination Examples:
  - type name = expression;

Constants

- Values that cannot change are called constants or non-changeable values
- 2 different types of constants are "literals" and "named" constants
- A literal constant is any number or character that is in the computer
- A named constant is a declared and assigned variable that cannot change
- Named constants go inside the program file but outside the main method
- Naming Example:

    - public static final int MAX_DAYS = 31
- Constant names are to be in all capital letters and separated by underscores
- This is done for readability and maintainability
- We also use named constants to change multiple values by changing only one variable

Expressions

- Expressions are two or more operands connected by one or more operators
- Example:
    - operand operator operand
    - n1 + n2
- Mathematical operators: **+, -, *, /**
- The type of variable is important to the result of expressions
- If two values of different values are combined the result is higher in precision that the original values, therefore, requiring a higher type of variable for the result
- The **modulus** operator: **"%"** only gets the remainder of that division
- Precedence rules in order: **parentheses**, **unary operators**, **multiplication** and **division** and **modulus**, **addition** and **subtraction**

Unary Operators

- Unary operators consist of just one operator and one operand
- Signed operators
- Increment and decrement operators are meant to increase a number by one or decrease a number by one: ++, --
- You can put these operators in prefix (++count) or postfix (count++); these determine what is calculated first
- Postfix doesn't change the value of the variable until after the entire expression is complete; it only uses the operator the next time the variable is used

Strings

- A **String** is a sequence of zero or more characters enclosed in double quotes
- Strings are **not** considered to be a **primitive** data type
- Declaration: String variable = "Hello!"; The capital "S" in "String" is how to call that data type
- **Concatenation** connects two or more Strings together
- **Escape characters** are told do something that it was not originally meant to do
- To use escape characters you use the backslash (\)
- Example:
    - System.out.println("he said, \"That was awesome!\"") = he said, "That was awesome!"
- **New line** character - (\n) - this will start the text on a new line
- **Indent** character - (\t) - puts an indent in the text
- **Carriage return character** - (\r) - returns to the current line
- To display a "%" in a printf() you have to use "%%"

Type Cast

- Example:
    - double distance = 0.7;
    - int points = (int)distance;
- "(int)" is a type cast which converts distance to a smaller value

Keyboard and Screen I/O

- Screen output
- println() - automatically adds a new line character
- print() - does not automatically add a new line
- printf() - the "f" stands for "format" and allows us to display to the user the value of our strings
    - The format specifier always starts with a "%" sign
    - **f** is for a floating-point value, **d** is for and integer value

- Examples:
  - System.out.printf("My age is **%5d**", **age**); - This will put a minimum of 5 spaces
  - System.out.printf("PI is equal to **%.2f**", **PI**); - This will limit the answer to 2 decimal points
  - System.out.printf("My age is **%d** Yay! **%d**", **age**, **value**); - multiple specifiers
  - System.out.printf("Balance is **%7.2f**", **balance**); - minimum width is 7 spaces, decimal spaces limited to 2

- Keyboard input
- Examples:
  - keyboard.nextInt();
  - keyboard.nextDouble();
  - keyboard.nextFloat();
  - keyboard.next(); - can only store one word
  - keyboard.nextLine(); - can store entire lines

Documentation and Style

- Meaningful variable names
- Comments - notes written within the program to help a programmer understand what's going on
  - Examples:
    - // This is a single line comment (use two forward slashes)
    - /* This is a multiline comment */ everything between those operators is considered a comment
- Indentation - stay consistent
- Named constants
- Program comments - The description at the top of the program

CS 202-1 Chapter 3

September 27, 2018 8:23 AM

Objectives

- Learn to use if statements
- Understand how to use boolean expressions
- Use logical operators
- Compare strings
- Indentation does not affect the execution of the code

Flow of Control: Branching Statements

- Algorithms - steps to solve a problem
- All algorithms have a certain order in which to solve problems
- We have been using sequence - one line after the other
- There is also selection and iteration
- A selection or branching statement chooses between to different options
  - The if-else statement
    - Always has true and false paths determined by conditions
    - "if" is true
    - "else" is false
    - All if statements are based on conditions
    - != - not equal to
    - Curly braces - required if the statement has more than one line of code
- **Indentation does not affect the execution of the code**
- Flowchart representations
  - The "diamond" or "decision" symbol represents the conditions of the program and has at least two possible solutions

How the IF Operates

- Syntax
  - There must be parentheses after the "if" statement
  - There should never be a semicolon after the parentheses of the "if" statement
  - There are never parentheses after the "else" statement
- Boolean expressions
  - Boolean expressions result in a true or false value
    - Simple expression:
      - If(value >= 7)
    - Boolean variable:
      - If(valueIsOkay)
    - Complex expression:
      - If(value >= 7 && valueIsOkay)
- Logical operators
  - Example:
    - "and" and "or" operators
    - And - **&&**
    - Or - **||**
    - Not - **!**
      - Example:
        - if (!(number >= min)) is the same as if (number < min)
        - if (!(number < max)) is the same as if (number >= max)
  - Short-circuit evaluation - the answer is calculated as soon as it's known
  - Order of precedence:
    - And - **&&**
    - Or - **||**

        - Not - **!**
- Comparing strings
    - Never use equality comparison with floating point numbers because there may be other decimal points involved
    - String examples:
        - String name1;
        - String name2;
        - if (name1.equals("hello there"))
    - Method examples:
        - .equals
        - .equalsIgnoreCase
        - keyboard.next().toLowerCase()
        - keyboard.next().toUpperCase()

Reading In a Single Character

- The "0" in the .charAt(0) is the position at which the method finds the single character
- 0 is the first position, 1 is the second position, 2 is the third position, and so on...
- Example:
    - char myLetter
    - myLetter = keyboard.next().charAt(0);
    - myLetter = keyboard.next().toUpperCase().charAt(0);
        - .charAt(0) - is a method

Nested If-else Statements

- If statements inside of if statements are called nested if-else statements
    - There is a risk of creating a dangling else statement if you are not careful
    - Use curly braces to avoid this problem
- A multi-branch statement is when an If statement is inside of an else statement
    - Technically this is the same as a nested if-else statement
- For the sake of clarity, always use curly braces
- If-else statements allow us to create more than two possible paths
- The "true" path is positive logic and the "false" path is negative logic

Additional Concepts

- Emergency exit
    - System.exit(0);
- Boolean input
    - ownCar = keyboard.nextBoolean();

The Switch Statement

- Switch statement - allows a controlling expression to take one of many possible paths
    - The data type must be either int, char, or String; cannot be anything else
    - Example:
        - switch (myNumber)
          {
          case label:
          Lines of code;
          break;
          }
    - Every label in the switch statement must be unique
- Any switch statement could be written as a nested if statement, but not all nested if statements can be written as a switch statement

CS 202-1 Chapter 4

October 25, 2018 8:10 AM

Looping Statements

- The portion of a statement or a group of statements is called a **loop**
- The portion of the loop that is repeated inside the statement is called the **loop body**
- Each repetition of the loop statement is called an **iteration**
- First thing to know is **where** you need your loop in the program
- Second thing you should know is **when** to **stop** the loop
- Three categories of loops:
  - Indefinite loop - when the loop begins and it doesn't know exactly when it will stop
    - While loop
    - Do-while loop
  - Definite loop - when the loop begins it already knows how many times it will loop until it stops
    - For-loop
  - Infinite loop - loops that never stop; usually considered to be a logic error
- Important parts of the loop:
  - **Control variable** - A value that **changes** for each iteration of the loop
  - **Sentinel value** - tells the loop when to stop; never changes value
  - **Test condition** - the entire expression including the control variable and sentinel value; must evaluate as a boolean expression

The WHILE loop

- The while loop **repeats** a set of actions as long as the controlling boolean expression (test condition) is true
  - Better definition:
    - The while loop repeats a set of actions **indefinitely** and tests the condition before those actions
    - **Indefinite, pre-test** loop
  - It is possible for the while loop to never be executed if the conditions are false the first time
- Two important items:
  - The control variable must be initialized **before** it reaches the test condition
  - The control variable must be modified **inside** of the loop

The DO-WHILE Loop

- The do-while loop **repeats** a set of actions as long as the controlling boolean expression (test condition) is true
  - Better definition:
    - The body of the do-while loop is always executed at least once
    - Tests the condition after the body is executed
    - **Indefinite, post-test** loop
    - **Least used** loop
- **The control variable must be initialized**
  - **This can happen inside the loop body**
  - **Most likely, the same statement used to initialize the variable also modifies the control variable in the condition**

The FOR Loop

- Better definition:
  - A **counter-controlled loop** designed to repeat a specific number of times (p.219)
  - Essentially a while loop
  - Used for definite situations
  - **Definite**, **pre-test** loop
- How it works
  - The flowchart is the same as a while loop
  - The **control variable** must be initialized as the first expression in parentheses
    - for (--> **x = 0** <--; x <= 3; x++)
  - **Sentinel value:**
    - for (x = 0; --> **x <= 3** <--; x++)
  - **Modification:**
    - for (x = 0; x <= 3; --> **x++** <--)

- If the control variable is declared inside the for loop then it cannot be used outside the loop
  - Declaring and initializing the control variable:
    - for (**int x = 0**; ;)

Additional Information

- Common programming mistakes
  - Forgetting to initialize the loop control variable
  - Forgetting the modify the loop control variable
    - Almost always results in an infinite loop
      - To break out of an infinite loop:
        - **Windows - ctrl + c**
        - **Linux - ctrl + x**
    - Make sure to modify the correct way
  - Off-by-one error - repeats the statement one too many times or one too few times
  - Misplaced semicolons
- **Regression testing**
  - Re-testing the program to make sure that the changes made don't break the program
- Special statements
  - Statements that should only be used sparingly:
    - **break;**
      - This will immediately end the loop and continue to the end of the program
    - **continue;**
      - This will skip the rest of the loop body and will continue the loop iterations
  - **goto** statement
    - Java doesn't have it
- **Important parts of the loop:**
  - Control variable - modified after each loop iteration
  - Sentinel value - the boolean expression that ends the loop
  - Test condition - the entire loop condition including the control variable and sentinel value

CS 202-1 Chapter 5

November 26, 2018 8:05 AM

Methods

- Definition of a Method
  - A structure containing code
  - An action performed within a program or by an object
    - A group of code that collectively performs an action within a program
  - Can be **called**, **executed,** and **returned** in that order
  - **May return a value**
  - Various names
    - Method - Java uses this name
    - Function
    - Procedure
- Reasons for using methods
  - Methods allow for **abstraction**
    - The suppression of irrelevant details
  - Methods are good for **reusability**
    - prevents re-writing code multiple times
    - Better efficiency when writing the program
  - Methods are good for **manageability**
    - This means to break up the code into separate pieces
    - Allows for easy **development** of code
    - Allows for easy **maintenance** of code
  - Methods allow for **testing**
    - Allows for easy development of code
    - Allows for easy maintenance of code
  - Methods allow for **delegation**
    - Allows multiple people to work on separate methods and still work together
  - Methods are **professional**
    - Every program uses multiple methods
    - Every programmer codes using multiple methods
- Understanding methods
  - There are two different types of methods
    - Methods that return a **single value**
      - value = keyboard.nextInt(); - where keyboard.nextInt() is the method that is being returned with a value
    - **Void** methods
      - Methods that do not return a value
        - System.out.println("hello"); - where println() does not return a value
  - Method planning
    - Involves deciding what methods need to be written before actually writing them
    - Basic computer operations
      - **Input**, **processing**, and **output**
    - Hierarchy charts
      - Takes the biggest picture and separates it into smaller pieces

Method Characteristics

- Methods exist in two parts
  - The **definition** and the **call**
- The body of a method is always **required** to have **curly braces**
- Method **definition/declaration**
  - ```
    public static int getFavNumber()
    {
        return 42;
    }
    ```
- The keyword "**static**" is required in the declaration of the method (the book does not show this)
- Methods always **return** but they may return a value
- **Local variables** only exist in the method they are created in or the **scope** of that variable
- **Global variables** are evil
- **Parameters** are a way to share values
  - The **names** of the variables don't have to match but the **data types** do have to match
- Only one value can be returned in methods
- **Actual parameters** are the variables in the parentheses of the methods inside the main method
- **Formal parameters** are the variables in the parentheses of the methods outside the main method

Documentation

- **Documentation** - is like a user manual that shows diagrams, instructions, and examples to a product; support material to help both the user and the program
  - **Flowcharts** are considered as documentation
    ```

- **In-code comments** are considered as documentation
- Documentation = comments
- The main contributor to code-level documentation is not comments; it is good programming style
    - Good identifier names

CS 202-1 Extras

December 5, 2018 8:17 AM

Object-Oriented Concepts

- **Object-oriented programming** - a programming methodology
  - An **object** is like a mini program, they have their own **variables** and **methods**
  - Objects **interact** with one another
- Terminology
  - In order to have an object you first need a **class**
  - **Class** - a definition or a **blueprint** of what an object should look like
  - A class doesn't define a specific object, it just describes the general characteristics of a **type** of object
  - An object is a specific **instance** of a class; considered to be **instantiation**
  - **Attributes** - **private**; are **variables** specific to a class; still not considered to be global variables
  - **Behaviours** - **public**; **methods** inside of the class; methods defined for the class

Structure

- **Structure** - mode of building, construction, or organization
- Striving for excellence
- **Macho code** - fancy code or solution that solves the problem but is done in a complex or unnecessary way