LESSON 1 - THE BASICS

java is a high level programming language. developed in 1990s

by Sun Microsystems and is now maintained by Oracle

java is platform independent. this means you only need to

write a program once to be able to run it anywhere

the source extension for java files is “.java”

most java apps are distributed using a java archive file; it ends in “.jar”

jdk = java development kit

jar = java archiving tool

jdb = java debugging tool

javac = java compiler

jre = java runtime environment

jvm = java virtual machine

the jvm is an execution environment for java applications - it interprets compiled

java binary code (aka byte code) and translates it into machine code

the jvm enables java to be platform independent

the jvm allocates memory needed by the java program

ide = integrated development environments

api = application programming interface

in java, every line of code that runes needs to be in a class

the main method is executed first

code pertaining to the method/class is enclosed in curly braces

every line must end with a semicolon ( ; ) or a bace (  {  ), (  }  )

comments are done using two forward slashes, like this:

//this is a comment

multi-line comments spans multiple lines. they look like this:

/\*

This Is A

Multi-Line

Comment

\*/

java does not support nested multi-line comments, but you can put

single line comments in multi-line comments

java doc comments are done by enclosing comments in a /\*\* and \*/

variables store data for processing

a variable is given a name such as area, height, etc.

variables have types such as integer, double, string, etc.

int: store integers (whole numbers)

double: for floating-point numbers (decimal numbers)

String: store texts (words)

you create variables like this:

String name = “David”;

int x = 123;

int y = 420.69;

you cannot store text in an int or a double

ints do not store decimals. so 15 divided by 4 is 3, not 3.75. use a

double when dealing with large numbers or decimals

more examples of variables include char and boolean

char letter1 = ‘A’; // char stores a single letter. also, you have to use single quotes to use it

boolean online = true // boolean has only two possible values, true or false

you can use operands to manipulate variables. these include:

addition ( + )

subtraction ( - )

multiplication ( \* )

division ( / )

module ( % )

the module operator finds the remainder. for example 5 % 2 = 1. this is because

2 fits into 5, 2 times with 1 left over. therefore the answer is 1

the increment and decrement operators provide a compact way to increase or decrease

the value of a variable by 1. so instead of this:

x = x + 1;

x = x - 1;

you can do the following:

x++; // to increase

x--; // to decrease

these operators can come before or after (prefix, and postfix). for example:

++x; // this is prefix

x++; // this is postfix

in prefix, the value of ‘x’ is incremented FIRST, and in prefix it is AFTER

you can even apply this shorthand notation to the ( = ) operator:

( += )

( -= )

( \*= )

( /= )

( %= )

to visualize, this:

x = x + y

becomes this:

x += y

to join two or more strings together, you need to concatenate them with the ( + ) operator. i.e.:

System.out.println(firstName + lastName);

to get user input you need to import the Scanner library like this:

import java.util.Scanner; // this goes at the very top of your code

then, to use the Scanner class, you need to create an instance of it like this:

Scanner myVar = new Scanner(System.in);

to read the input, you can use any one of the following methods:

Read a byte - nextByte()

Read a short - nextShort()

Read an int - nextInt()

Read a long - nextLong()

Read a float - nextFloat()

Read a double - nextDouble()

Read a boolean - nextBoolean()

Read a line - nextLine()

Read a word - next()

LESSON 2 - CONDITIONAL STATEMENTS

conditional statements are used to perform actions based on conditions/requirements/criteria. i.e.

if statement (if this is true, then do this)

while loops (if a number is 5 then continue to do this)

for loops (same as while)

do while loops (same as while, but WILL execute once, even if condition is false)

you must use comparison operators to help evaluate the condition. i.e.

< (less than)

> (greater than)

!= (not equal to)

== (equal to)

<= (less than or equal to)

>= (greater than or equal to)

the syntax for if statement:

if (condition) { // Code goes here }

syntax for while loops:

while (condition) { // Code }

for loops syntax:

for (initialization; condition; increment) { // Code }

you can even nest statements if you need to. for example:

if (condition) {

if (condition) {

//Code

}

}

so the code above will only execute if both conditions are met. you can rewrite the code above

using logical operators such as ( && ). i.e.:

if (condition1 && condition2) { // code }

more logical operators include ( || ) and ( ?: )

( || ) this is OR

( ?: ) this is called ternary operator and can replace the if statement

( && ) this is AND

if statements can be joined with an else or else if statement. i.e.:

if (condition) {

//code

} else {

//other code

}

OR

if (condition1) {

//code

} else if (condition2) {

//other code

} else if (condition3) {

//some code

} else { // last code }

instead of an if else statement, you can even use a switch statement. it tests a variable for equality against

a list of values. i.e.

switch (expression) {

case value1:

//Code

break;

case value2:

//Code

break;

case value3:

//Code

break;

default:

//Code

//no break needed here b/c this is the last line of code

}

the break is needed to jump out of the switch statement. always put a break when using this

use a break and continue statement to change the execution of the loop’s flow. a break statement

will cause you to jump out of the loop, even if the condition is not met. a continue statement will cause

the loop to skip out on the task. so if u print numbers from 1 to 10, but you don’t wanna print 5, then use

a continue statement to skip it.

LESSON 3 – ARRAYS

an array is a collection of variables of the same type. They are used when you need to store a list of values

such as numbers, strings, etc. instead of declaring 20 different variables, you can store them in an array

you can declare an array like this:

int[] arr = new int[5]; // replace the 5 with how many values you want it to store

arr[0] = 23; // array counting starts at 0

to call upon a number stored in the array, call using its location. i.e.

arr[0]

more array examples:

String[] names = {“Bob”, “David”, “Jack”}

There are many commands that come with arrays. For instance you can access the number of elements stored

In the array. i.e.:

arr.length

because arrays are useful, the enhanced for loop was created to make iterating an array simple and efficient. Use this because it eliminates the possibility of bugs and makes the code easier to read. i.e.

int [] primes = {2, 3, 5, 7}

for (int t: primes) {

System.out.println(t);

}

multi-dimensional arrays contain other than 1 array. Example of a 2D array:

int [][] sample = {{1,2,3},{4,5,6}};

you are not limited to 1D array or even 2D array, you can have 3D, 4D, etc. arrays

LESSON 4 - CLASSES AND OBJECTS

Java uses Object-Oriented Programming (OOP); a programming style that is intended to make thinking about programming closer to thinking about the real world. In OOP, each object is an independent unit with a unique identity – similar to objects in the real world.

Objects have characteristics, which are used to describe them. i.e. A car can be red or blue, a mug can be empty or full. These characteristics are called attributes. An attribute describes the current state of an object. Objects also have behaviors. A car moves, a person walks, a dog barks, etc.

In OOP, each object has three dimensions: identity, attributes, and behavior. i.e.

Identity: John Smith

Attributes: Age, Height, Weight, Etc.

Behavior: Eat, Sleep, Walk, Jump, Laugh, Etc.

A class describes what the object will be. They are like the blueprints for an object. Classes define attributes and behavior.

A method is a collection of statements that are grouped together to perform an operation. i.e. Printing out an error message

The return keyword can be used in methods to return a value. You can have a method defined to return the sum of two values

Void means that the method doesn’t not return a value

Static means that you don’t have to instantiate it to use it

The main method must always be static and void

Access modifiers are used to set the level of access

For classes, you can use default and public. The default modifier makes the class accessible by classes in the same package, and the public modifier makes the class accessible by any other class

For attributes and methods, you can use default, public, protected and private. Protected provides the same access as the default access modifier, with the addition that subclasses can access protected methods and variables of the superclass. Private makes the only accessible within the declared class itself

Getters and setters are used to protect data. The get method returns the value, and the set method sets it

The keyword “this” is used to refer to the current object

Constructors are special methods invoked when an object is created and are used to initialize them. It can be used to provide initial values for object attributes

A constructor name must be same as its class name and have no explicit return type

Value types short, int, long, float, double, boolean, and char. These data types store values assigned to them.

Java passes by value NOT reference

The math class allows you to use the following methods:

Math.abs (Returns the absolute value)

Math.ceil (Rounds down)

Math.floor (Rounds up)

Math.max (Returns the maximum value)

Math.min (Returns the minimum value)

Math.pow (Calculates the power)

Math.sqrt (Square root)

Math.sin (Calculates sin)

Math.cos (Calculates cos)

Use the final keyword to mark a variable as constant so it will never change. i.e.

Public static final double PI = 3.14159;

Use packages and classes to organize your code

LESSON 5 - MORE ON CLASSES

There are 4 core concepts in OOP: encapsulation, inheritance, polymorphism, and abstraction.

Encapsulation is used to mask details from users. The variables of one class will be hidden from the variables of other classes. This is also called data hiding. This is done by declaring the class’ variables as private and using getters and setters to manipulate.

Inheritance is when one class acquires the properties (methods and variables) of another class. With inheritance, the information is placed in a more manageable, hierarchical order.

The class inheriting the properties of another class is the subclass, and the class whose properties are inherited is the superclass. i.e.:

class Dog extends Animal {…}

Dog is the subclass and Animal is the superclass (Use “extends” to inherit another class)

When the subclass inherits from the superclass, it inherits all non-private methods and variables

Polymorphism is one method, different implementations. So you’ll have a superclass named Animal and then subclasses like dog, cats, fish, turtle, etc. And each subclass will have its own its behaviors (methods) and attributes

Method overriding is known as runtime polymorphism.

Data abstraction provides the outside world with only essential information. i.e. When you hear of a book, you know what a book is but you don’t know the exact logistics of said book

An Enum is a special type used to define collections of constants. They define variables that represent a fixed set. i.e. Military rankings

Use a wildcard to import all classes in a package