COP3502 Module 04 Review

byte num = 127;

num++; **// OVERFLOW**

num will become = **-**128 because **+**128 was too large to put into a byte





**Primitive Data Types: Allow for DIRECT comparison**

Table

Description automatically generated

integers—



Integer overflow/underflow: When you try to put a number that is too large/too small into a type of integer that cannot hold it, you get integer overflow/underflow.

double x = 0.1 + 0.1; // 0.2

double y = 0.1 \* 2; // 0.2

**x == y 🡪 FALSE**

A picture containing table

Description automatically generated

floating point —

* Floating Point (FP) numbers give better *precision*, but sometimes lower *accuracy*
* When using FP numbers, you can get some funny results… Ex] 5.0 / 0.0 = Infinity

Solution:

final threshold = 0.0001;

**Math.abs(x-y) < threshold**

* FP numbers are difficult to compare directly using “==”



boolean —



character —

* A character is a **single symbol** (can be a letter, a number, a punctuation symbol, etc.)

Examples] **‘A’**, **‘a’**, **‘$’**, **‘ ’** (space), **‘\n’** (escape sequence 🡪 \n is regarded as ONE character)

So, you might be wondering… if a computer reads everything in binary (1’s and 0’s) ***how does a computer read characters like letters and symbols??***

**Answer:** Characters are first converted to their equivalent decimal number values via the **ASCII encoding table**. Then those decimal numbers are converted into binary! Ex] **‘A’ 🡪 65** 🡪 1000001

**Constants: Variables with values that DO NOT change**

* CAPITALIZE all constant variable names and **declare using final keyword**
* DO NOT try to change the value of your constant – you will get an error

**Type Conversion (Implicit and Explicit)**

**final** double PI = 3.141596;

double circleArea = PI \* r \* r;

PI = 4.0; **// ERROR**

Implicit type conversion- conversion happens “automatically”. You don’t have to tell your compiler to convert between types… it just does it.

Ex]

Explicit type conversion- you must explicitly tell your compiler the type that you want to convert to.

Ex]

String message = “My favorite number is “ + 22;

System.out.println(message);



**integer 🡪 FP num =**

**FP num 🡪 integer =**

**(possible loss of info)**

Type conversion!

Ouput: Double number is 12.5589388

Integer number is 12

double num = 12.5589388;

System.out.println(“Double number is: ” + num);

System.out.println(“Integer number is: “ + **(int)** num);

Output: My favorite number is 22

COP3502 Module 04 Review

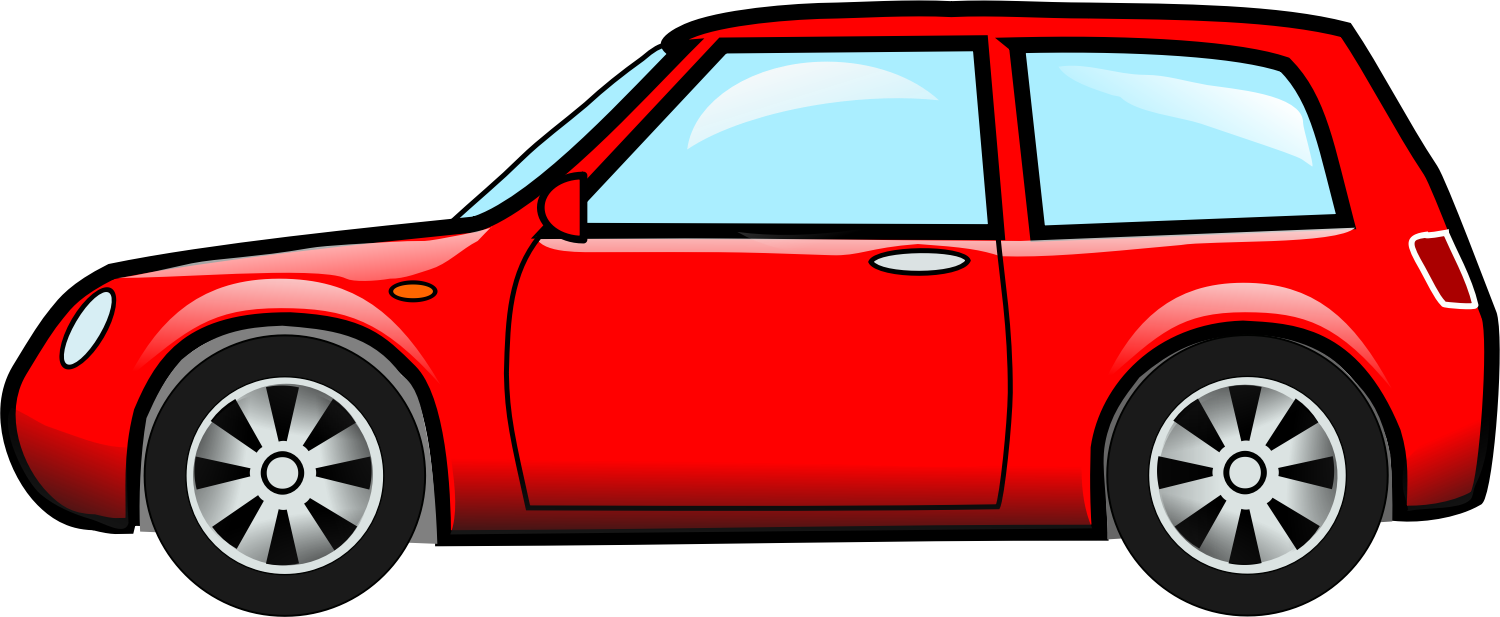
**Objects**







**Objects and Strings:**



Object- and object is an **instance of** a class

Examples] A car is an **instance of** a vehicle, so a car is an object

SpongeBob is an **instance of** a sea creature, so SpongeBob is an object

Objects have attributes and methods associated with them





Car:

Possible attributes:

* int numOfMiles
* String make
* String model

Possible Methods:

* driveFaster()
* takeTurn()

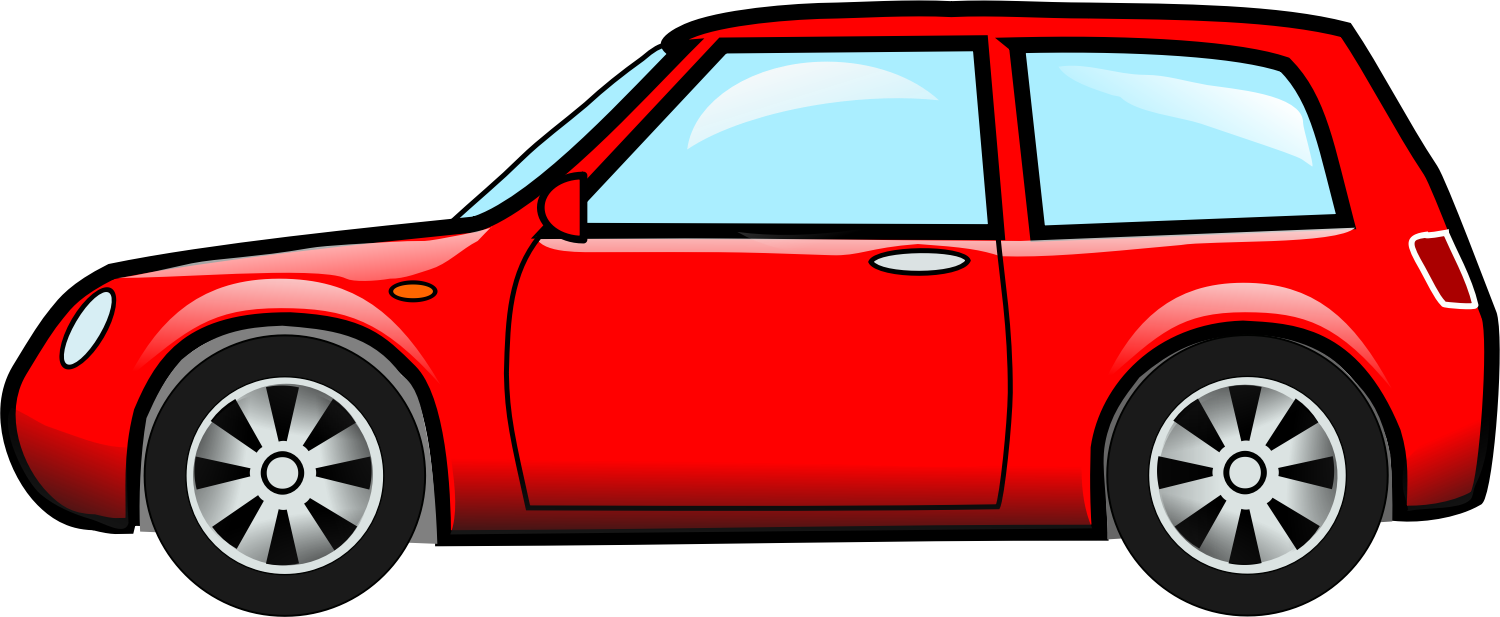
Spongebob:

Possible attributes:

* String friend = “Patrick”
* String house = “Pineapple”
* int numOfPets = 1

Possible Methods:

* makeCrabbyPatty()
* annoySquidward()







Object oriented programming- combines attributes and methods into an object

String- sequence of characters Ex] “Hello World!”

* Strings are IMMUTABLE – once created, they cannot be changed
* A String is NOT a primitive type it is a REFERENCE type which means:
  + - * + **YOU CAN NOT COMPARE THEM WITH “==”**

charAt() is an example of an **instance method** 🡪 it is called on by “message” which is an instance of the String class

* + - * + They hold reference to an OBJECT
* Individual characters in a string can be accessed via the function “charAt”

String message = “Hello!”;

Char letter = message.charAt(4); // letter = ‘o’

Instance methods- can only be called by an object (an instance of) the class

valueOf (in this situation) is an example of a **static method** 🡪it is not called on by an “object” of the class but rather by the class itself

Static methods- can be called without an object (an instance of) the class

String answer = String.valueOf(42); // answer = “42”

Ex]

When we want to have set of values that are non-numeric, we can use enumeration

enum WeekDay { Monday, Tuesday, Wednesday, Thursday, Friday }

Ex]