NYIT

**Fall 2021**

Homework No: 02

Title: Classes Objects

**Name:** Patade, Yash Arun

**Class ID#:**  11

**School ID#:** 1284979

**Course:** Java Networking

**Course ID:** CSCI 725

**Date:** 09/22/2021

Assignment Contents

[1) Question: Abstraction? 4](#_Toc83448684)

[(i) What does abstract refer to? What does it mean? 4](#_Toc83448685)

[2) What are classes? 5](#_Toc83448686)

[Description: 5](#_Toc83448687)

[3) Representing Objects. 6](#_Toc83448688)

[4) Homework: Classes 6](#_Toc83448689)

[(i) Give example of 3 objects and their common class representative. 6](#_Toc83448690)

[(ii) Give example of 3 boundary-stereotype classes. 7](#_Toc83448691)

[(1) Communication between Main menu and customer\_database. 7](#_Toc83448692)

[(2) Communication between Login page and Credentials Authentication. 7](#_Toc83448693)

[(3) Communication between Access Controller and Authorized Personnel. 7](#_Toc83448694)

[(iii) Give example of 3 entity-stereotype classes. 8](#_Toc83448695)

[(1) Professor teaches the student. 8](#_Toc83448696)

[(2) Student registering for course. 8](#_Toc83448697)

[(3) Customer maintaining login credentials 8](#_Toc83448698)

[5) In class, what gets inherited? 9](#_Toc83448699)

[6) Inheritance Hierarchy Super and Sub Classes. 9](#_Toc83448700)

[7) Modifications in C and C++/Java. 10](#_Toc83448701)

[8) this in constructors. 11](#_Toc83448702)

[9) Execution of MathLibraryExample.java 13](#_Toc83448703)

[10) Compiling MathLibraryExample.java 19](#_Toc83448704)

[11) Executing MathLibraryExample.java 19](#_Toc83448705)

[12) Output for MathLibraryExample.java 21](#_Toc83448706)

[13) Environment Variables. 25](#_Toc83448707)

[ Shell provides execution environment to programs that start via environment variables. 25](#_Toc83448708)

[14) Shell B – Local Environment Variables. 26](#_Toc83448709)

[ Individual shell environment may be customized by modification of existing shell variable values of by addition of new shell variables. 26](#_Toc83448710)

[15) System – Wide Environment Variable CLASSPATH 27](#_Toc83448711)

[16) Question: What sort of environment variable is PATH? 28](#_Toc83448712)

[ Standard environment variable. 28](#_Toc83448713)

[ Comment: Needed by the most of the programs that may run in the given operating environment OE i.e. OS. 28](#_Toc83448714)

[ PATH provides to shell ordered sequence of OSFS directories that have to be searched when executable file is needed to start a new program. 28](#_Toc83448715)

[17) Question: What sort of environment variable is variable X? 29](#_Toc83448716)

[ Variable X custom environment variable. 29](#_Toc83448717)

[ Needed maybe to support execution of some custom application program that would like to echo its value at some point at run time. 29](#_Toc83448718)

[18) What are PATH option value separators in Windows and Unix? 29](#_Toc83448719)

[ Windows uses character ‘;’ and UNIX uses character ‘:’ 29](#_Toc83448720)

[ Windows ‘:’ character is already used to separate storage volume or partition name ‘C’ from the root-directory symbol ‘\’ in “C:” string which means: “Root directory on OSFS volume names C.” 29](#_Toc83448721)

[References. 30](#_Toc83448722)

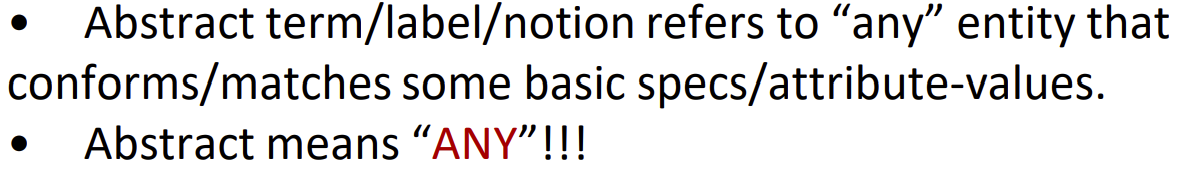
Work from [Microsoft PowerPoint - 725\_02\_Lecture\_01\_Classes\_Nootes\_\_\_\_\_\_\_.pptx (tfbor.com)](http://tfbor.com/02_725/02_ClassesObjects/725_02_Lecture_01_Abstraction_Classes.pdf)

# 1) Question: Abstraction?

## What does abstract refer to? What does it mean?



Answer)



# 2) What are classes?

## Description:

• There are many objects identified for any domain

• A class is a description of a group of objects with common properties (attributes), common behavior (operations), common relationships to other objects (associations and aggregations), and common semantics

– An object is an instance of a class

• A class is an abstraction in that it:

– Emphasizes relevant characteristics

– Suppresses other characteristics

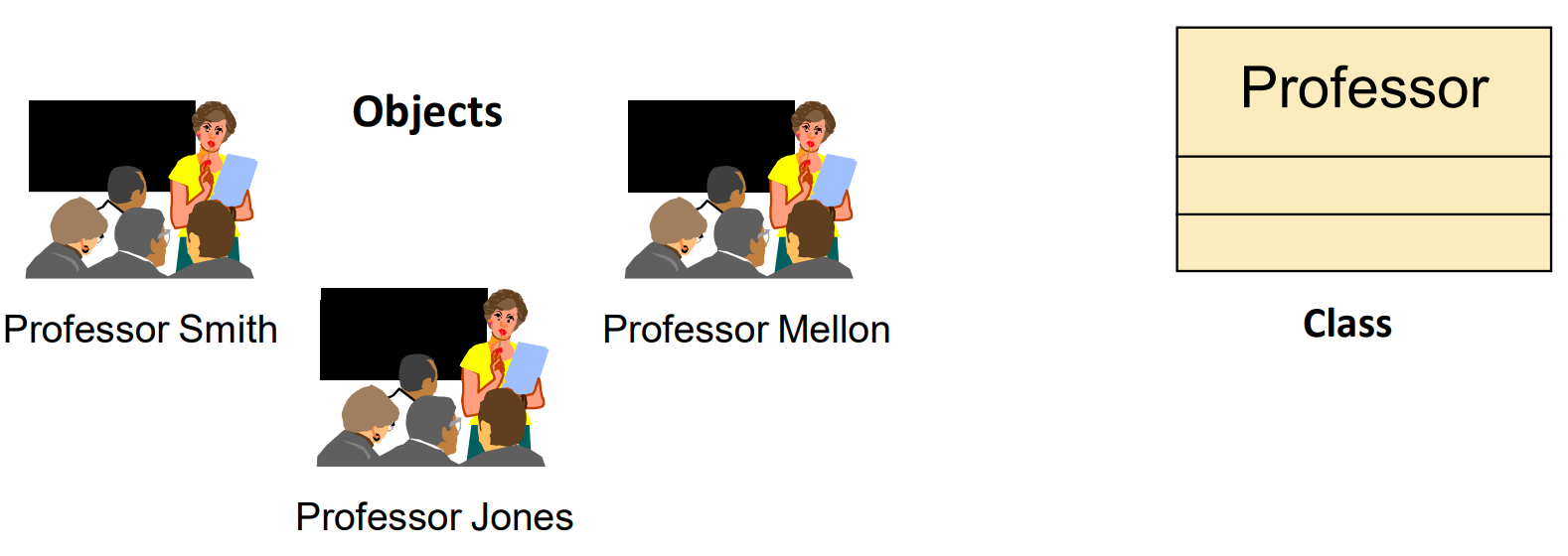
• Abstraction helps us deal with complexity

• A class is an abstract definition/description/model of an object

– It defines the structure and behavior of each object in the class.

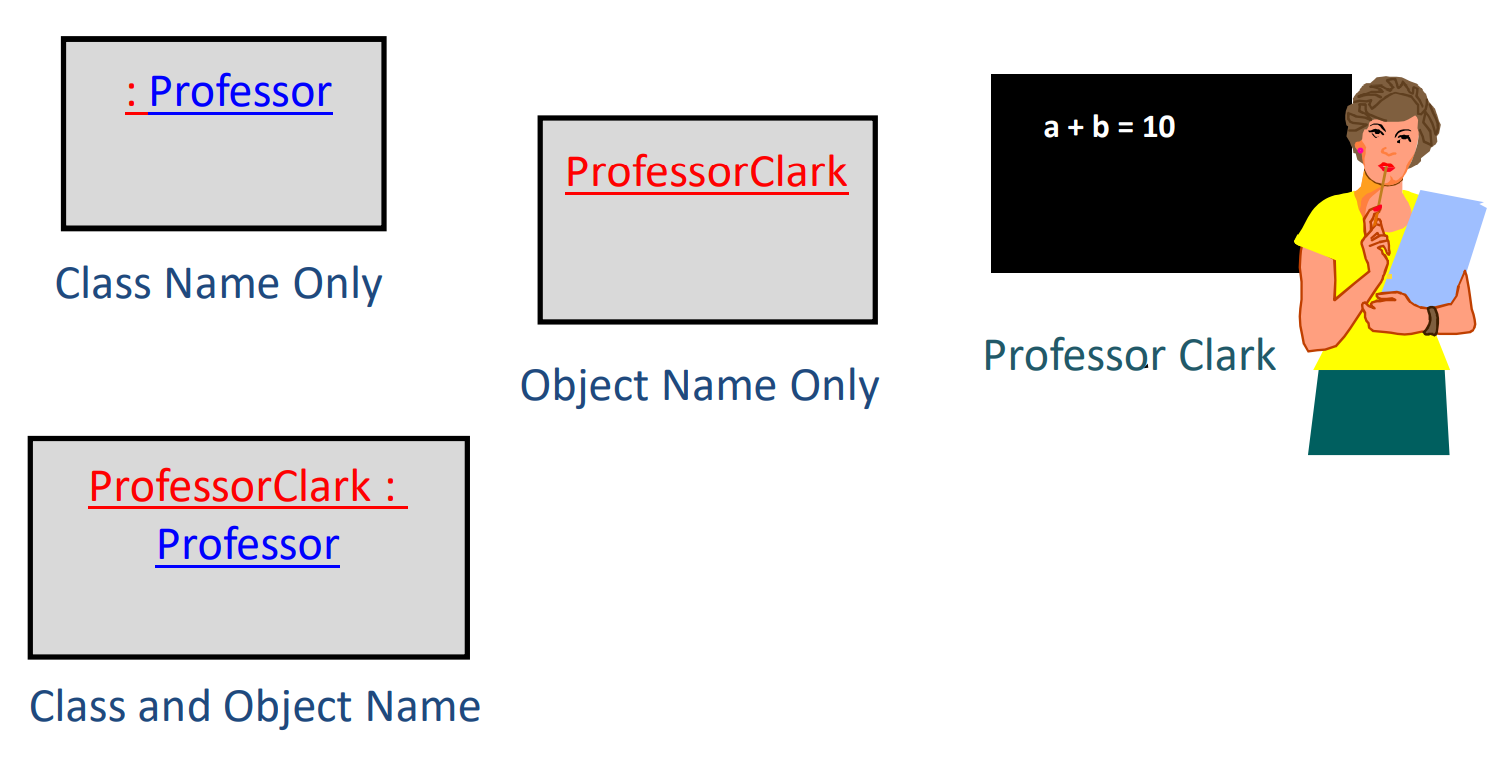
– It serves as a template for creating objects.

• Objects may be grouped into classes



# 3) Representing Objects.

• An object is represented in UML as rectangles with underlined object name/reference (Sometimes type/class-name is added).



# 4) Homework: Classes

## Give example of 3 objects and their common class representative.

Answer) An object has identity. Each object has a unique identity, even if its state is identical to that of another object.

class

car

Objects

- color

- make\_year

- model\_no

- price

Ford Tesla Toyota

## Give example of 3 boundary-stereotype classes.

### (1) Communication between Main menu and customer\_database.

**Cust\_database()**

**person\_01**

**person\_02**

**person\_03**

**Main Menu**

**Cust\_database()**

**Feature\_02()**

**Feature\_03()**

### (2) Communication between Login page and Credentials Authentication.

**Credentials**

**Authentication**

**Login Page**

**Check if correct:**

* **username**
* **password**

**Collect:**

* **username**
* **password**

### (3) Communication between Access Controller and Authorized Personnel.

**Authorized Personnel**

**Access Controller**

*is granted access respectively.*

**Check:**

* **credentials**
* **job title**
* **level of access**

## Give example of 3 entity-stereotype classes.

### (1) Professor teaches the student.

**Student**

**Professor**

**- student\_ID**

**- name**

teaches

**- emp\_ID**

**- name**

**- course\_ID**

### (2) Student registering for course.

**course**

**Student**

**- student\_ID**

**- name**

**- course\_ID**

**- course name**

**- course number**

registers

### (3) Customer maintaining login credentials

**- cust\_ID**

**- name**

**- amount paid**

**- contact info**

**- city**

maintains

**Customer**

**Login**

**- customer\_ID**

**- username**

**- password**

Work from [Microsoft PowerPoint - 725\_02\_Lecture\_02\_Inheritance\_Notes\_\_\_.pptx (tfbor.com)](http://tfbor.com/02_725/02_ClassesObjects/725_02_Lecture_02_Inheritance_notes.pdf)

# 5) In class, what gets inherited?

Answer)

• A subclass inherits its parent’s:

– Attributes

– Operations

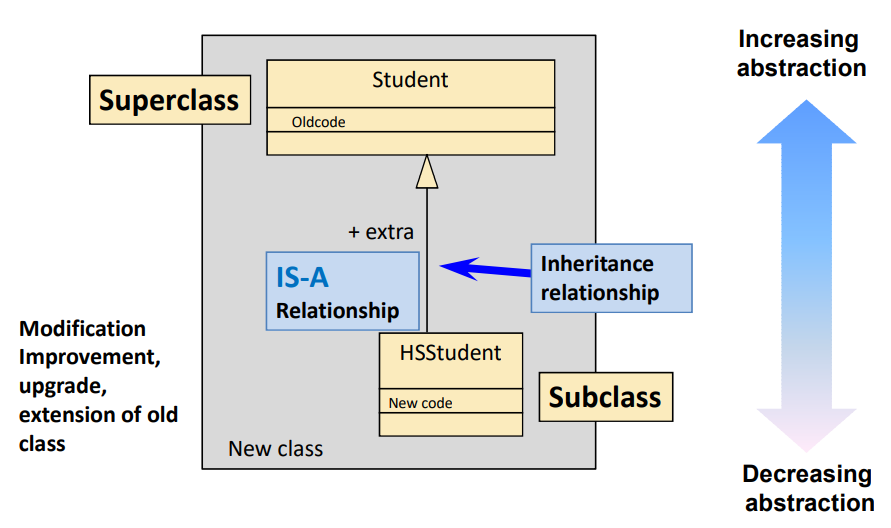
– Relationships

• A subclass may:

– Add additional attributes, operations, relationships

– Redefine inherited operations

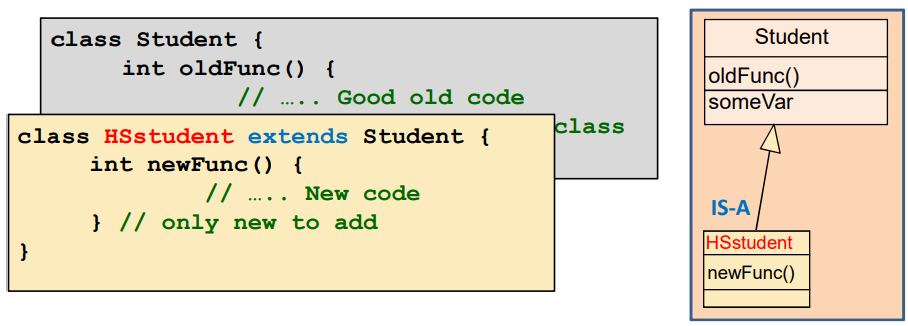
# 6) Inheritance Hierarchy Super and Sub Classes.



# 7) Modifications in C and C++/Java.

• In C we have to open Student program and modify to get HS student new program…. Many chances for bugs.

• In C++/Java we do not touch good old Student class, we just write separate addition to class **Student** and indicate that new combined class copies all form the old class Student using Java key word extends.



Work from [725\_02\_Homework\_02\_Constructor.docx (live.com)](https://view.officeapps.live.com/op/view.aspx?src=http%3A%2F%2Ftfbor.com%2F02_725%2F02_ClassesObjects%2Fhomework%2F725_02_Homework_02_Constructor.docx&wdOrigin=BROWSELINK)

# 8) this in constructors.

It is often the case that overloaded methods are essentially the same except that one supplies default values for some of the arguments. In this case, your code will be easier to read and maintain (though perhaps marginally slower) if you put all your logic in the method that takes the most arguments, and simply invoke that method from all its overloaded variants that merely fill in appropriate default values.

This technique should also be used when one method needs to convert from one type to another. For instance, one variant can convert a String to an int, then invoke the variant that takes the int as an argument.

This straight-forward for regular methods, but doesn't quite work for constructors because you can't simply write a method like this:

public Car(String licensePlate, double maxSpeed) {

Car(licensePlate, 0.0, maxSpeed);

}

Instead, to invoke another constructor in the same class from a constructor you use the keyword this like so:

public Car(String licensePlate, double maxSpeed) {

this(licensePlate, 0.0, maxSpeed);

}

Must this be the first line of the constructor?

For example,

public class Car {

private String licensePlate; // e.g. "New York A456 324"

private double speed; // kilometers per hour

private double maxSpeed; // kilometers per hour

// constructors

public Car(String licensePlate, double maxSpeed) {

this(licensePlate, 0.0, maxSpeed);

}

public Car(String licensePlate, double speed, double maxSpeed) {

this.licensePlate = licensePlate;

if (maxSpeed >= 0.0) {

this.maxSpeed = maxSpeed;

}

else {

maxSpeed = 0.0;

}

if (speed < 0.0) {

speed = 0.0;

}

if (speed <= maxSpeed) {

this.speed = speed;

}

else {

this.speed = maxSpeed;

}

}

// other methods...

}

This approach saves several lines of code. In also means that if you later need to change the constraints or other aspects of construction of cars, you only need to modify one method rather than two. This is not only easier; it gives bugs fewer opportunities to be introduced either through inconsistent modification of multiple methods or by changing one method but not others.

Work from [725\_02\_Homework\_03\_Math\_LibraryDemo\_2\_Programs.pdf (tfbor.com)](http://tfbor.com/02_725/02_ClassesObjects/homework/725_02_Homework_03_Math_LibraryDemo_2_Programs.pdf)

# 9) Execution of MathLibraryExample.java

**public class MathLibraryExample {**

**private static int i = 7;**

**private static int j = -9;**

**private static double x = 72.3;**

**private static double y = 0.34;**

**/\*\***

**\* Helper utility used to print a String to STDOUT.**

**\* @param s String that will be printed to STDOUT.**

**\*/**

**private static void prt(String s) {**

**System.out.println(s);**

**}**

**private static void prt() {**

**System.out.println();**

**}**

**/\*\***

**\* Helper utility used to print all variables used through out all**

**\* example Math methods.**

**\*/**

**private static void printVariables() {**

**prt("PRINT VARIABLES");**

**prt("===========================================");**

**prt(" (int) i = " + i);**

**prt(" (int) j = " + j);**

**prt(" (double) x = " + x);**

**prt(" (double) y = " + y);**

**prt();**

**}**

**/\*\***

**\* The Math library defines several useful constants.**

**\*/**

**private static void printMathConstants() {**

**prt("MATH CONSTANTS");**

**prt("===========================================");**

**prt(" Pi is " + Math.PI);**

**prt(" e is " + Math.E);**

**prt();**

**}**

**/\*\***

**\* The absolute value of a number is equal to the number if the number is**

**\* positive or zero and equal to the negative of the number if the number**

**\* is negative.**

**\*/**

**private static void doAbsoluteValues() {**

**prt("ABSOLUTE VALUE");**

**prt("===========================================");**

**prt(" |" + i + "| is " + Math.abs(i));**

**prt(" |" + j + "| is " + Math.abs(j));**

**prt(" |" + x + "| is " + Math.abs(x));**

**prt(" |" + y + "| is " + Math.abs(y));**

**prt();**

**}**

**/\*\***

**\* Round off a floating point number to the nearest integer with round().**

**\* <p>**

**\* The "ceiling" of a number is the smallest integer greater than or equal**

**\* to the number. Every integer is its own ceiling.**

**\* <p>**

**\* The "floor" of a number is the largest integer less than or equal to the**

**\* number. Every integer is its own floor.**

**\*/**

**private static void doTruncateRounding() {**

**prt("TRUNCATING AND ROUNDING FUNCTIONS");**

**prt("===========================================");**

**prt(" " + x + " is approximately " + Math.round(x));**

**prt(" " + y + " is approximately " + Math.round(y));**

**prt();**

**prt(" The ceiling of " + i + " is " + Math.ceil(i));**

**prt(" The ceiling of " + j + " is " + Math.ceil(j));**

**prt(" The ceiling of " + x + " is " + Math.ceil(x));**

**prt(" The ceiling of " + y + " is " + Math.ceil(y));**

**prt();**

**prt(" The floor of " + i + " is " + Math.floor(i));**

**prt(" The floor of " + j + " is " + Math.floor(j));**

**prt(" The floor of " + x + " is " + Math.floor(x));**

**prt(" The floor of " + y + " is " + Math.floor(y));**

**prt();**

**}**

**/\*\***

**\* Comparison operators. min() returns the smaller of the two arguments**

**\* (numbers) you pass it.**

**\* <p>**

**\* max() returns the the larger of the two arguments (numbers) you pass it.**

**\*/**

**private static void doComparisonOperators() {**

**prt("COMPARISON OPERATORS");**

**prt("===========================================");**

**prt(" min(" + i + ", " + j + ") is " + Math.min(i,j));**

**prt(" min(" + x + ", " + y + ") is " + Math.min(x,y));**

**prt(" min(" + i + ", " + x + ") is " + Math.min(i,x));**

**prt(" min(" + y + ", " + j + ") is " + Math.min(y,j));**

**prt();**

**prt(" max(" + i + ", " + j + ") is " + Math.max(i,j));**

**prt(" max(" + x + ", " + y + ") is " + Math.max(x,y));**

**prt(" max(" + i + ", " + x + ") is " + Math.max(i,x));**

**prt(" max(" + y + ", " + j + ") is " + Math.max(y,j));**

**prt();**

**}**

**/\*\***

**\* Trigonometric Methods. All arguments are given in radians.**

**\*/**

**private static void doTrigonometricMethods() {**

**prt("TRIGONOMETRIC METHODS");**

**prt("===========================================");**

**// Convert a 45 degree angle to radians**

**double angle = 45.0 \* 2.0 \* Math.PI/360.0;**

**prt(" cos(" + angle + ") is " + Math.cos(angle));**

**prt(" sin(" + angle + ") is " + Math.sin(angle));**

**// Inverse Trigonometric methods. All values are returned as radians**

**double value = 0.707;**

**prt(" acos(" + value + ") is " + Math.acos(value));**

**prt(" asin(" + value + ") is " + Math.asin(value));**

**prt(" atan(" + value + ") is " + Math.atan(value));**

**prt();**

**}**

**/\*\***

**\* Exponential and Logarithmic Methods.**

**\*/**

**private static void doExponentialLogarithmicMethods() {**

**prt("EXPONENTIAL AND LOGARITHMIC METHODS");**

**prt("===========================================");**

**// exp(a) returns e (2.71828...) raised to the power of a.**

**prt(" exp(1.0) is " + Math.exp(1.0));**

**prt(" exp(10.0) is " + Math.exp(10.0));**

**prt(" exp(0.0) is " + Math.exp(0.0));**

**prt();**

**// log(a) returns the natural logarithm (base e) of a.**

**prt(" log(1.0) is " + Math.log(1.0));**

**prt(" log(10.0) is " + Math.log(10.0));**

**prt(" log(Math.E) is " + Math.log(Math.E));**

**prt();**

**}**

**/\*\***

**\* Using Power Method. pow(x, y) returns the x raised to the yth power.**

**\*/**

**private static void doPower() {**

**prt("POWER METHOD");**

**prt("===========================================");**

**prt(" pow(2.0, 2.0) is " + Math.pow(2.0,2.0));**

**prt(" pow(10.0, 3.5) is " + Math.pow(10.0,3.5));**

**prt(" pow(8, -1) is " + Math.pow(8,-1));**

**prt();**

**}**

**/\*\***

**\* Using Power Method. qrt(x) returns the square root of x.**

**\*/**

**private static void doSquareRoot() {**

**prt("SQUARE ROOT METHOD");**

**prt("===========================================");**

**for (i=0; i < 10; i++) {**

**prt(" The square root of " + i + " is " + Math.sqrt(i));**

**}**

**prt();**

**}**

**/\*\***

**\* Random Numbers. Java provides a Random method that returns a**

**\* pseudo-random number between 0.0 and 1.0.**

**\*/**

**private static void doRandomNumbers() {**

**prt("RANDOM NUMBERS");**

**prt("===========================================");**

**prt(" Random number: " + Math.random());**

**prt(" Random number: " + Math.random());**

**prt();**

**}**

**public static void main(String[] args) {**

**prt();**

**printVariables();**

**printMathConstants();**

**doAbsoluteValues();**

**doTruncateRounding();**

**doComparisonOperators();**

**doTrigonometricMethods();**

**doExponentialLogarithmicMethods();**

**doPower();**

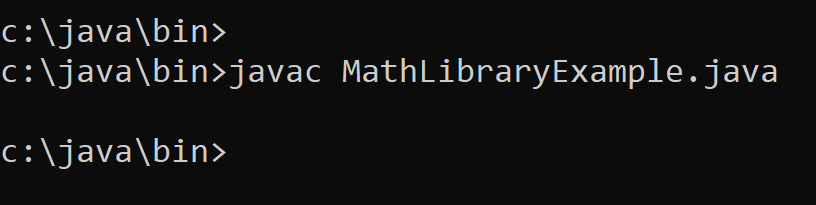
**doSquareRoot();**

**doRandomNumbers();**

**}**

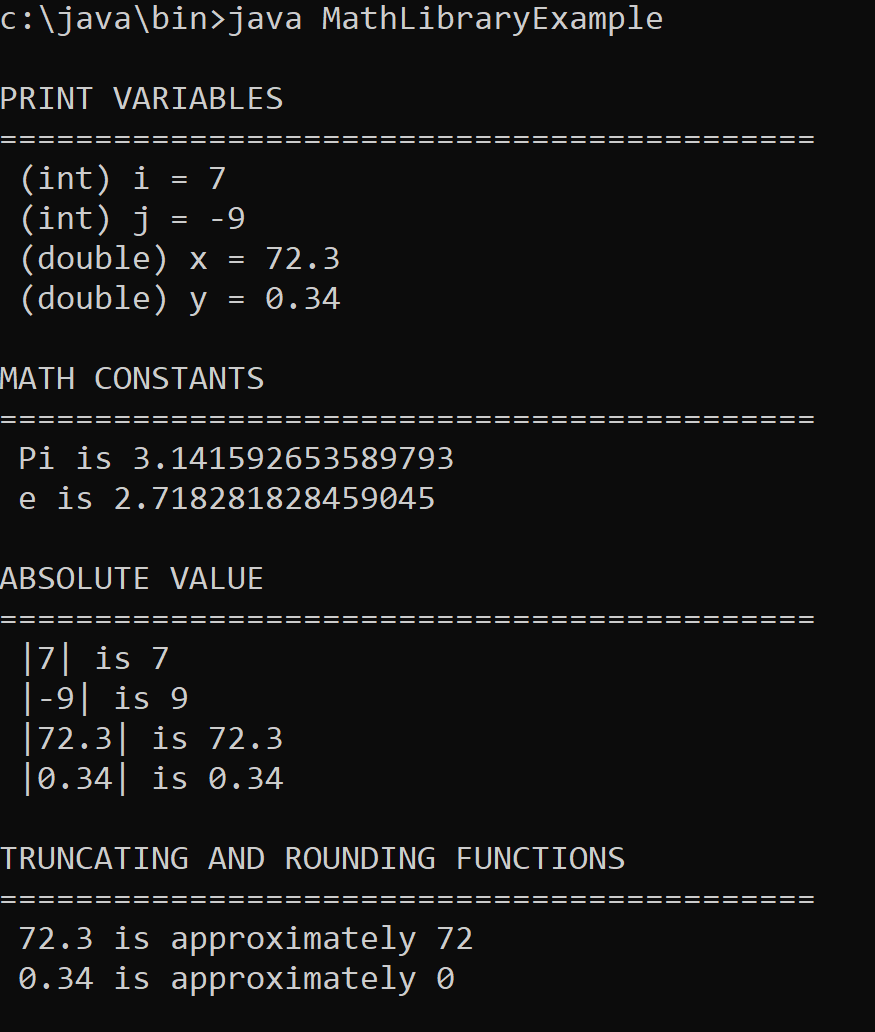
**}**

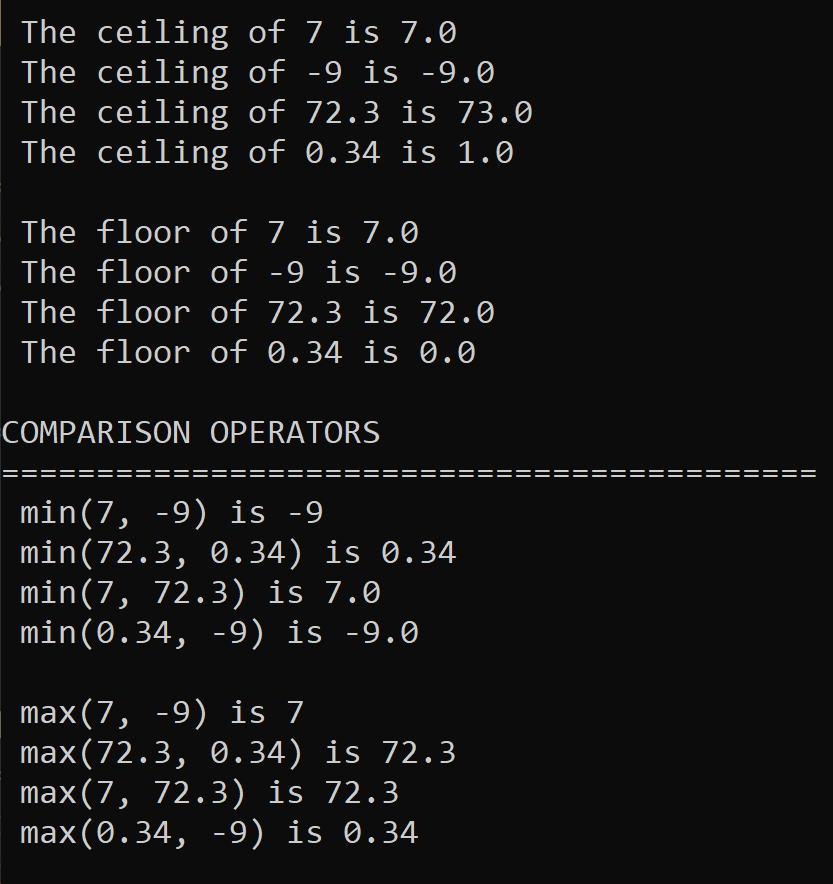
# 10) Compiling MathLibraryExample.java

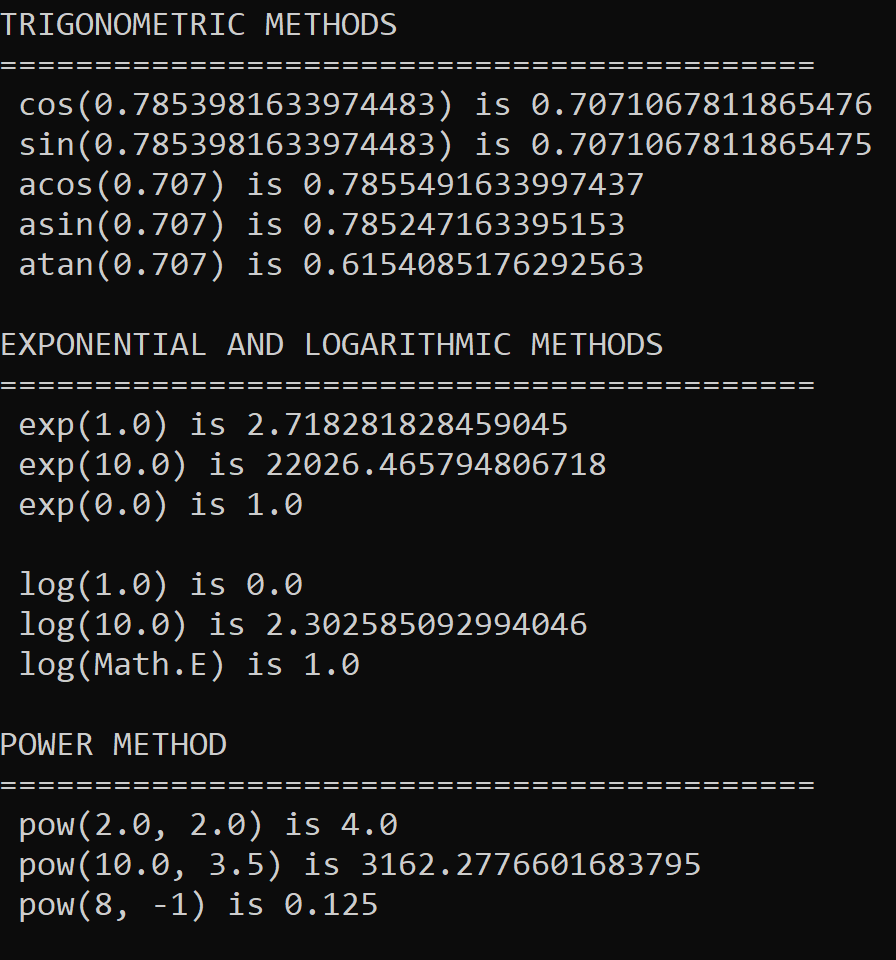


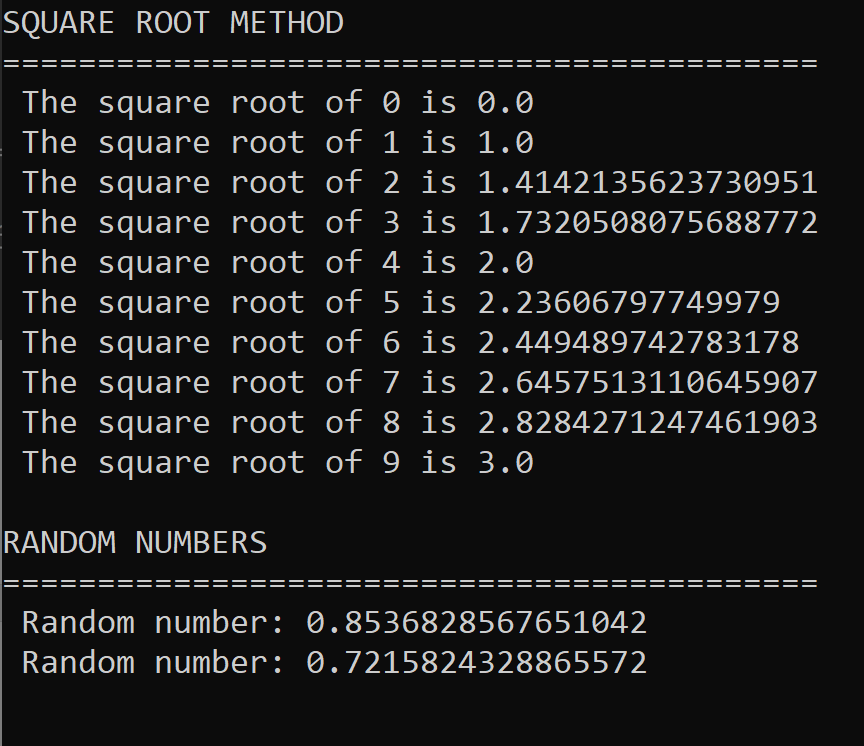
No errors.

# 11) Executing MathLibraryExample.java









# 12) Output for MathLibraryExample.java

PRINT VARIABLES

===========================================

(int) i = 7

(int) j = -9

(double) x = 72.3

(double) y = 0.34

MATH CONSTANTS

===========================================

Pi is 3.141592653589793

e is 2.718281828459045

ABSOLUTE VALUE

===========================================

|7| is 7

|-9| is 9

|72.3| is 72.3

|0.34| is 0.34

TRUNCATING AND ROUNDING FUNCTIONS

===========================================

72.3 is approximately 72

0.34 is approximately 0

The ceiling of 7 is 7.0

The ceiling of -9 is -9.0

The ceiling of 72.3 is 73.0

The ceiling of 0.34 is 1.0

The floor of 7 is 7.0

The floor of -9 is -9.0

The floor of 72.3 is 72.0

The floor of 0.34 is 0.0

COMPARISON OPERATORS

===========================================

min(7, -9) is -9

min(72.3, 0.34) is 0.34

min(7, 72.3) is 7.0

min(0.34, -9) is -9.0

max(7, -9) is 7

max(72.3, 0.34) is 72.3

max(7, 72.3) is 72.3

max(0.34, -9) is 0.34

TRIGONOMETRIC METHODS

===========================================

cos(0.7853981633974483) is 0.7071067811865476

sin(0.7853981633974483) is 0.7071067811865475

acos(0.707) is 0.7855491633997437

asin(0.707) is 0.785247163395153

atan(0.707) is 0.6154085176292563

EXPONENTIAL AND LOGARITHMIC METHODS

===========================================

exp(1.0) is 2.718281828459045

exp(10.0) is 22026.465794806718

exp(0.0) is 1.0

log(1.0) is 0.0

log(10.0) is 2.302585092994046

log(Math.E) is 1.0

POWER METHOD

===========================================

pow(2.0, 2.0) is 4.0

pow(10.0, 3.5) is 3162.2776601683795

pow(8, -1) is 0.125

SQUARE ROOT METHOD

===========================================

The square root of 0 is 0.0

The square root of 1 is 1.0

The square root of 2 is 1.4142135623730951

The square root of 3 is 1.7320508075688772

The square root of 4 is 2.0

The square root of 5 is 2.23606797749979

The square root of 6 is 2.449489742783178

The square root of 7 is 2.6457513110645907

The square root of 8 is 2.8284271247461903

The square root of 9 is 3.0

RANDOM NUMBERS

===========================================

Random number: 0.8536828567651042

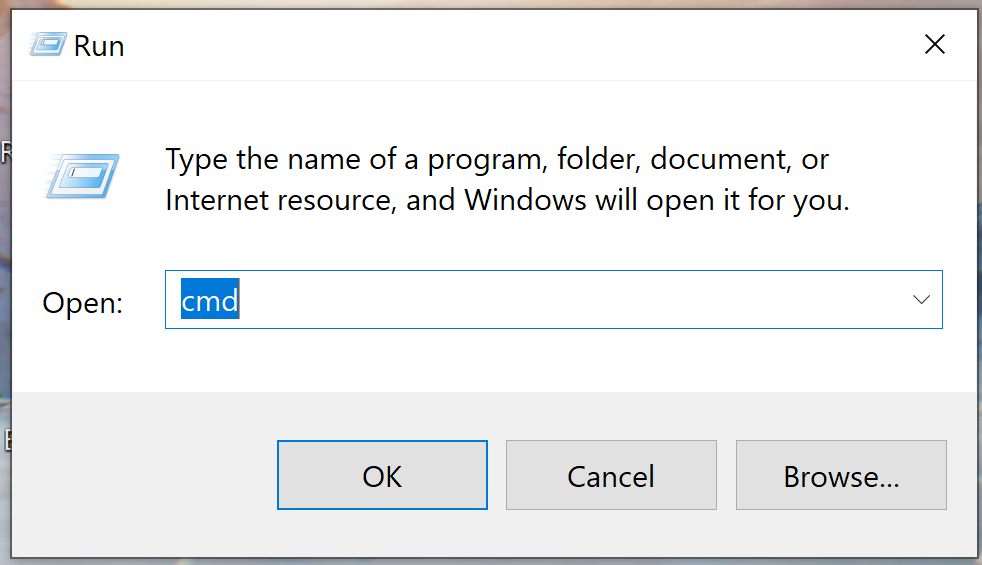
Random number: 0.7215824328865572

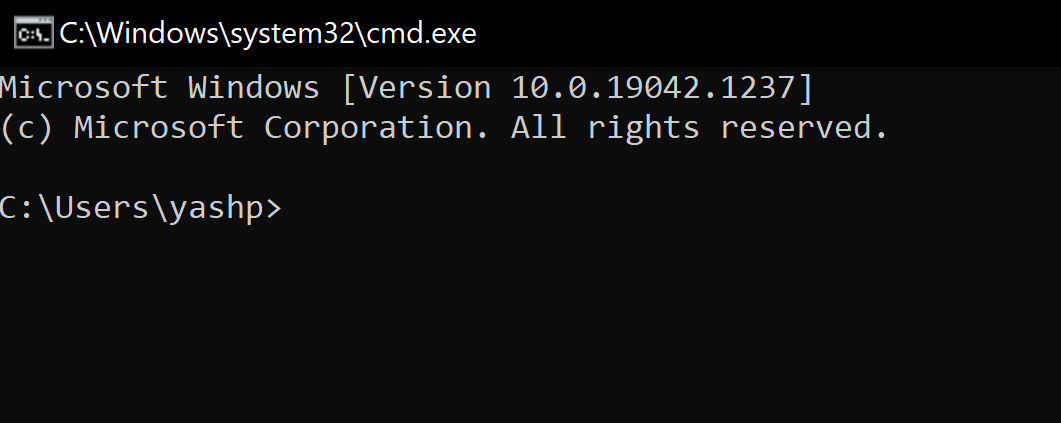
Work from [Microsoft PowerPoint - 325\_06\_EnvironmentVariables.ppt [Compatibility Mode] (tfbor.com)](http://tfbor.com/02_725/02_ClassesObjects/lab/_725_02_Exercise_01_EnvironmentVariables.pdf)

# 13) Environment Variables.

Shell A – Local Environment Variables

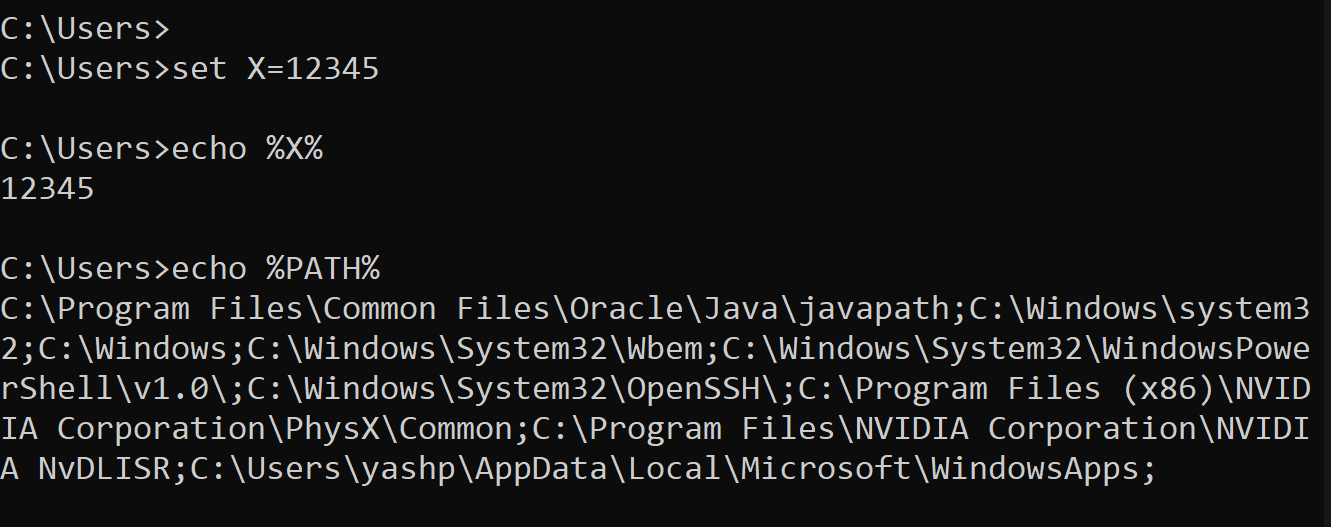
## Shell provides execution environment to programs that start via environment variables.

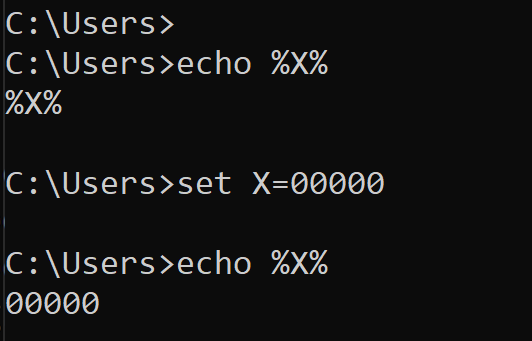




# 14) Shell B – Local Environment Variables.

## Individual shell environment may be customized by modification of existing shell variable values of by addition of new shell variables.

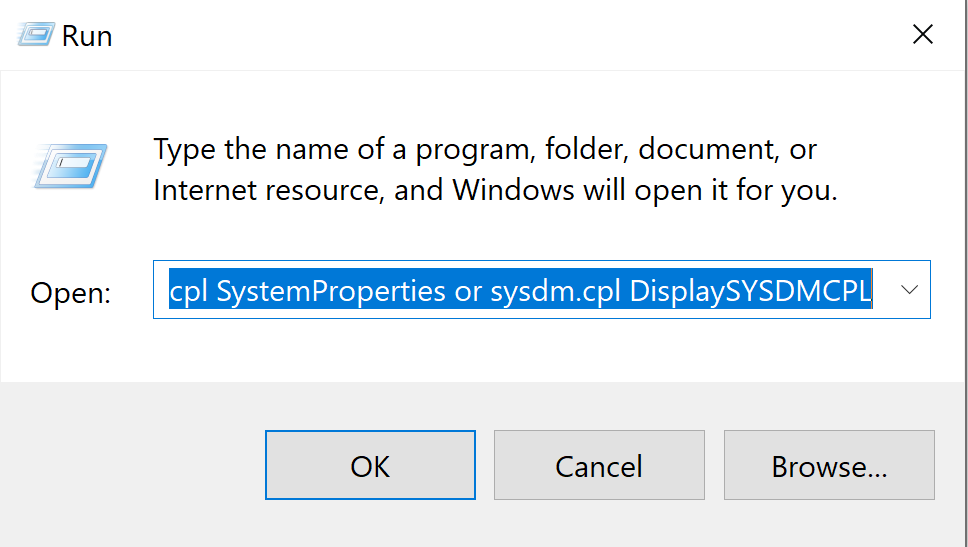


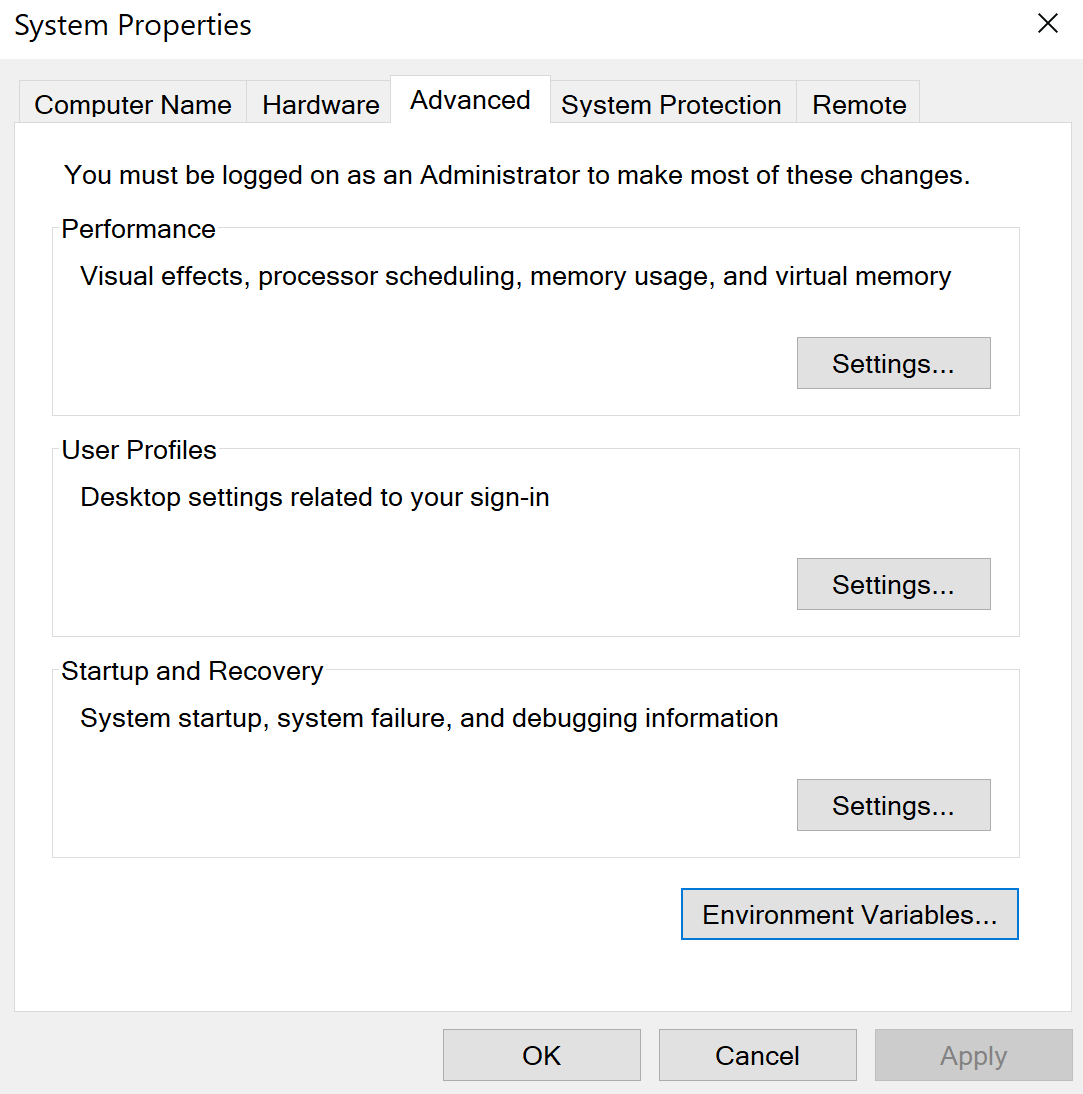


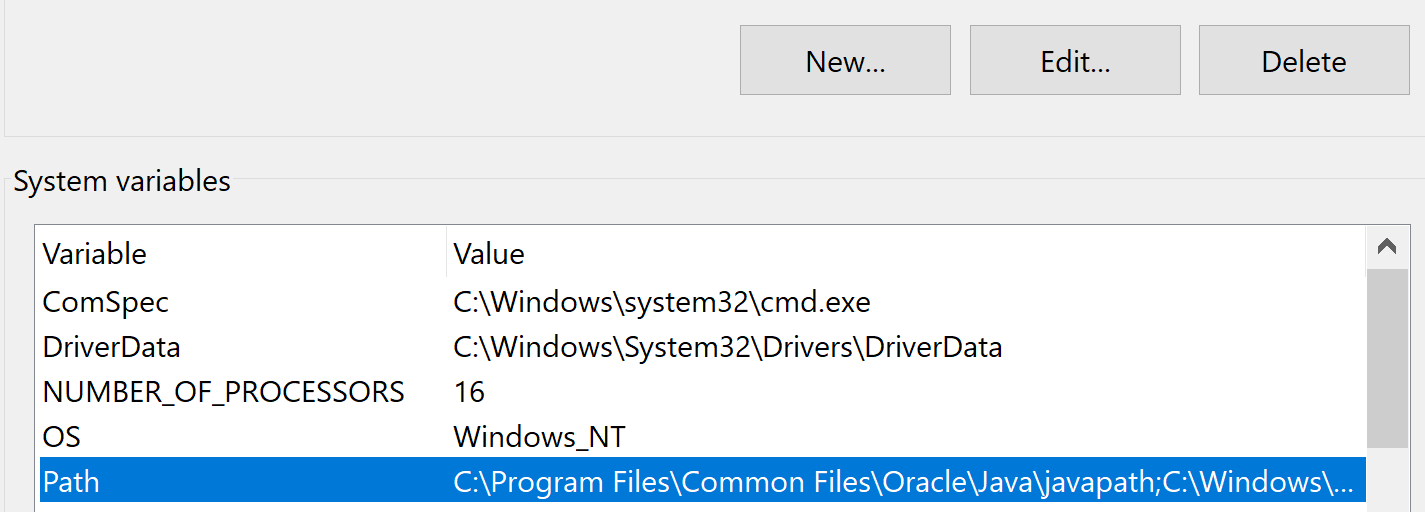
# 15) System – Wide Environment Variable CLASSPATH

Input following command in Run:

sysdm.cpl SystemProperties or sysdm.cpl DisplaySYSDMCPL







# 16) Question: What sort of environment variable is PATH?

Answer)

## Standard environment variable.

## Comment: Needed by the most of the programs that may run in the given operating environment OE i.e. OS.

## PATH provides to shell ordered sequence of OSFS directories that have to be searched when executable file is needed to start a new program.

# 17) Question: What sort of environment variable is variable X?

Answer)

## Variable X custom environment variable.

## Needed maybe to support execution of some custom application program that would like to echo its value at some point at run time.

# 18) What are PATH option value separators in Windows and Unix?

Answer)

## Windows uses character ‘;’ and UNIX uses character ‘:’

## Windows ‘:’ character is already used to separate storage volume or partition name ‘C’ from the root-directory symbol ‘\’ in “C:\” string which means: “Root directory on OSFS volume names C.”

# References.

1. [Index of /02\_725/02\_ClassesObjects (tfbor.com)](http://tfbor.com/02_725/02_ClassesObjects/)
2. [Microsoft PowerPoint - 725\_02\_Lecture\_01\_Classes\_Nootes\_\_\_\_\_\_\_.pptx (tfbor.com)](http://tfbor.com/02_725/02_ClassesObjects/725_02_Lecture_01_Abstraction_Classes.pdf)
3. [Microsoft PowerPoint - 725\_02\_Lecture\_02\_Inheritance\_Notes\_\_\_.pptx (tfbor.com)](http://tfbor.com/02_725/02_ClassesObjects/725_02_Lecture_02_Inheritance_notes.pdf)
4. [725\_02\_Homework\_03\_Math\_LibraryDemo\_2\_Programs.pdf (tfbor.com)](http://tfbor.com/02_725/02_ClassesObjects/homework/725_02_Homework_03_Math_LibraryDemo_2_Programs.pdf)
5. [Microsoft PowerPoint - 325\_06\_EnvironmentVariables.ppt [Compatibility Mode] (tfbor.com)](http://tfbor.com/02_725/02_ClassesObjects/lab/_725_02_Exercise_01_EnvironmentVariables.pdf)
6. [725\_02\_Homework\_02\_Constructor.docx (live.com)](https://view.officeapps.live.com/op/view.aspx?src=http%3A%2F%2Ftfbor.com%2F02_725%2F02_ClassesObjects%2Fhomework%2F725_02_Homework_02_Constructor.docx&wdOrigin=BROWSELINK)
7. [Entity Relationship Diagrams with draw.io - draw.io (drawio-app.com)](https://drawio-app.com/entity-relationship-diagrams-with-draw-io/)
8. [In UML class diagrams, what are Boundary Classes, Control Classes, and Entity Classes? - Stack Overflow](https://stackoverflow.com/questions/683825/in-uml-class-diagrams-what-are-boundary-classes-control-classes-and-entity-cl)