//Movie Classes (objects for age categories that offer discounts to movie ticket price)

if ($\_age>55) {

$this->\_ageCateg=$senior;

} elseif ($\_age>18) {

$this->\_ageCateg=$adult;

} elseif ($\_age>5) {

$this->\_ageCateg=$kid;

} else {

$this->\_ageCateg=$toddler;

}

}

public function setTktPrice ()

{

$this->\_age = $age;

}

//Now instantiate the objects for each of the “types” of ages for the price schedule

$senior = new MovieGoer ();

$adult = new MovieGoer ();

$kid = new MovieGoer ();

$toddler = new MovieGoer ();

$senior->echo “That will be $8 please”

$adult->echo “That will be $10 please”

$kid->echo “That will be $5 please”

$toddler->echo “Toddlers under age 5 are free to enter”

//An ALTERNATIVE DESIGN may need to specify the amount of the ticket price as a variable, especially if the program was going to be used for anything like calculating the bundle of ticket purchases e.g. or else chnging the dicounts offered or allowing discounts to apply to multiple groups etc. However given the instructions of the assignment the purpose I interpreted it to mean was simply separating objects and classes for each of the group and display the movie ticket price...at which point let the group attending the movie figure out how much the total is for what they owe.

//Still a very fun project and should be something that I can put to use too soon I hope.

/\*

NOTES:

More helpful examples to try out for the OOP stuff

Objects

<?php

$myPage = new Html;

// Creates an instance of the Html class

$myPage->printLn('This is some text');

// Prints "This is some text&lt;br /&gt;"

?>

INHERITANCE AND EXTENDING CLASSES (probably good for another module/topic)

//CLASS CONSTRUCTOR (IN PHP)...LESSON FOCUS

<?php

class Person

{

private $\_name;

private $\_age;

function \_\_construct($name, $age = 0)

{

if (!is\_int($age))

{

throw new Exception("Cannot assign non integer value to integer field, 'Age'");

}

$this->\_age = $age;

$this->\_name = $name;

}

}

?>

//CONSTRUCTOR OVERLOADING…..TBD

<?php

class BankAccount{

private $accountNumber;

private $totalBalance;

public function \_\_construct(){

}

public function \_\_construct($accountNo){

$this->accountNumber = $accountNo;

}

public function \_\_construct($accountNo, $initialAmount){

$this->accountNumber = $accountNo;

$this->totalBalance = $initialAmount;

}

}

?>

//ENCAPSULATING…..asking for user input and then validating it...useful for above code

public function setAge($age) {

if (!is\_int($age))

throw new Exception(“Cannot assign non integer value to integer field, ‘Age’”);

$this->\_age = $age;

}

\*/

{

abstract public function get\_name();

abstract public function set\_name($name);

public $phone\_number; //phone number is a property of a Contact

public function \_\_toString() //function acts on inputs to format the results using conditional loop and different operations

{

$s = "" . $this->get\_name();

if ($this->phone\_number) {

if (count($s) > 0) {

$s .= ": ";

} else {

$s .= "Someone's Phone Number: ";

}

$s .= $this->phone\_number;

}

return $s;

}

}

//The class extends BaseContact to persons and organizations

//The construct() function call sets default values for first and last name of this class which can be set then at any time with or without the object operation, using the new keyword.

class PersonContact extends BaseContact

{

public $first\_name;

//public $middle\_name;

public $last\_name;

public function \_\_construct($first\_name = null, $last\_name = null)

{

$this->first\_name = $first\_name;

$this->last\_name = $last\_name;

}

public function get\_name() //a straightforward simple function that returns first and last name with a space in between

{

return $this->first\_name . " " . $this->last\_name;

}

public function set\_name($name) //this function uses explode() for user input to propery assign name to two separate entities

{

$split\_name = explode(" ", $name, 3); //Changed this to 3 in order to shorten the 3 names down to 2 for consistency

$length = count($split\_name);

$rv = true;

if ($length === 0) {

$rv = false;

} elseif ($length === 1) {

$this->first\_name = $this->last\_name = $split\_name[0]; //here is where the array is created for PersonContact class objects

} else {

$this->first\_name = $split\_name[0];

// $this->middle\_name = $split\_name[1]; //I added this line to make the middle name undefined

$this->last\_name = $split\_name[1];

}

return $rv;

}

}

//A third class for Organizations that acts similar to the previous extended class but even simpler since no array is created for the name

class OrganizationContact extends BaseContact

{

public $name;

public function \_\_construct($name = null)

{

$this->name = $name; //every time the $this -> operator is called is when objects automatically call contructor() and object is created with default properties if no other is specified

}

public function get\_name()

{

return $this->name;

}

public function set\_name($name)

{

$this->name = $name;

}

}