PHP / MySQL

PHP Forms - $_GET Function

- > The built-in $_GET function is used to collect values from a form sent with method="get".
- > Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send (max. 2000 characters).

PHP Forms - $_GET Function

```
<form action="ex01.php" method="get">
        Name: <input type="text" name="fname">
        Age: <input type="text" name="age">
        <input type="submit" name="submit" value="GO">
</form>
```

Notice how the URL carries the information after the file name.

```
Welcome <?php echo $_GET["fname"]; ?>. <BR>
You are <?php echo $_GET["age"]; ?> years old!
```

PHP Forms - $_GET Function

- The “ex01.php" file can now use the $_GET function to collect form data (the names of the form fields will automatically be the keys in the $_GET array)

```
Welcome <?php echo $_GET["fname"]; ?>. <BR>
You are <?php echo $_GET["age"]; ?> years old!
```

PHP Forms - $_GET

- When using method="get" in HTML forms, all variable names & values are displayed in the URL.
- This method should not be used when sending passwords or other sensitive information!
- However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- URLs over 2,000 characters will not work in the most popular web browsers.
- Most webservers have a limit of 8192 bytes (8KB), which is usually configurable.

PHP Forms - $_POST

- The built-in $_POST function is used to collect values from a form sent with method="post".
- Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- Note: However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the **post_max_size** in the php.ini file).

PHP Forms - $_POST Function

```
<form action="ex02.php" method="post">
        Name: <input type="text" name="fname">
        Age: <input type="text" name="age">
        <input type="submit" name="submit" value="GO">
</form>
```

And here is what the code of ex02.php might look like:

```
Welcome <?php echo htmlspecialchars($_POST["fname"]); ?>. <BR>
You are <?php echo (int)$_POST["age"]; ?> years old!
```

PHP Forms - $_POST Function

- Apart from **htmlspecialchars()** and, it should be obvious what this does. **htmlspecialchars()** makes sure any characters that are special in html are properly encoded so people can't inject HTML tags or Javascript into your page.

```
$str = "This is some <b>bold</b> text.";
echo htmlspecialchars($str);
```

```
This is some &lt;b&gt;bold&lt;/b&gt; text.
```

- **(int)** Since we know age is a number, we can just convert it to an integer which will automatically get rid of any stray characters.

PHP Forms - $_POST

- When to use **method="post"**?
- Information sent from a form with the **POST** method is invisible to others and has no limits on the amount of information to send.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

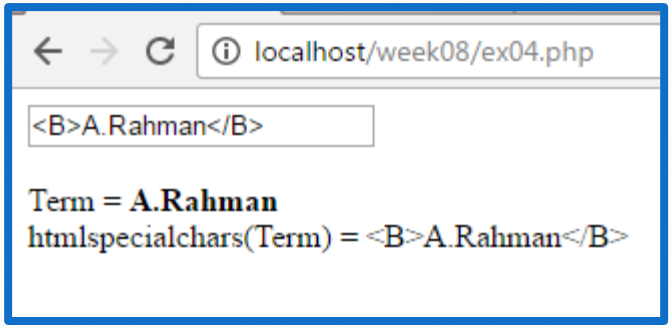
PHP - Forms

```
<?php
 if (isset($_POST["submit"]))
     echo "<h2>You clicked Submit!</h2>";
 if (isset($_POST["cancel"]))
     echo "<h2>You clicked Cancel!</h2>";
?>
```

```
<form action="ex03.php" method="post">
    <input type="submit" name="submit"
value="Submit">
    <input type="submit" name="cancel"
value="Cancel">
</form>
```

PHP - Forms

```
<form action="ex04.php" method="post">
    <input type="text" name="sterm"
      value="<?php if(isset($_REQUEST["sterm"]))
      echo $_REQUEST["sterm"]; ?>">
</form>
<BR>
<?php
if (isset($_REQUEST["sterm"])) {
      $term=$_REQUEST["sterm"];
      echo "Term = ".$term."<BR>";
      echo "htmlspecialchars(Term) =
      ".htmlspecialchars($term)."<BR>";
}?>
```



MySQL Connectivity

- **`mysql_connect()`**
  - The **`mysql_connect()`** function opens a non-persistent MySQL connection.
  - This function returns the connection on success, or FALSE and an error on failure. You can hide the error output by adding an '@' in front of the function name.
- Syntax
  - **`mysql_connect(server,user,pwd,newlink,client flag)`**

Persistent Connectins!

```
$link = mysqli_connect("$mysql_server", "$mysql_user",
"$mysql_pw", "$mysql_db");
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
```

- Persistent connection support was introduced in PHP 5.3 for the mysqli extension.
- Persistent connections are for Oracle, in which making a new connection is much slower.
- In MySQL, making a connection is fast. There should be no need for persistent connections with MySQL.

MySQL Connectivity

Parameter	Description
server	Specifies the server to connect to
user	Specifies the username to log in with.
pwd	Specifies the password to log in with.
newlink	If a second call is made to mysql_connect() with the same arguments, no new connection will be established; instead, the identifier of the already opened connection will be returned
clientflag	•MYSQL_CLIENT_SSL - Use SSL encryption •MYSQL_CLIENT_COMPRESS - Use compression protocol

MySQL Connectivity

- Example:

```
<?php
  $con =
  mysql_connect("localhost","mysql_user","mysq
  l_pwd");
  if (!$con){
    die('Could not connect: `.mysql_error());
  }
  echo 'Connected successfully';
  mysql_close($con);
?>
```

MySQL Connectivity

- **`mysql_close()`**
  - The **`mysql_close()`** function closes a non-persistent MySQL connection.
  - This function returns TRUE on success, or FALSE on failure.
- Syntax:
  - **`mysql_close(connection)`**

Parameter	Description
connection	Specifies the MySQL connection to close. If not specified, the last connection opened by **`mysql_connect()`** is used.

MySQL Connectivity

- **`mysql_select_db()`**
  - The mysql_select_db() function sets the active MySQL database.
  - This function returns TRUE on success, or FALSE on failure.
- Syntax:
  - **`mysql_select_db(database,connection)`**

Parameter	Description
database	Required. Specifies the database to select.
connection	Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

MySQL Connectivity

```
<?php
  $con = mysql_connect("localhost", “root",
                                “admin");
  if (!$con){
    die('Could not connect: '. mysql_error());
  }
  $db_selected = mysql_select_db("test_db",
                                        $con);
  if (!$db_selected){
    die ("Can\'t use test_db : “.
                                mysql_error());
  }
  mysql_close($con);
?>
```

MySQL Connectivity

- **`mysql_query()`**
  - The **`mysql_query()`** function executes a query on a MySQL database.
  - This function returns the query handle for SELECT queries, TRUE/FALSE for other queries, or FALSE on failure.
- Syntax
  - **`mysql_query(query,connection)`**

Parameter	Description
query	Required. Specifies the SQL query to send (should not end with a semicolon)
connection	Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql_connect() or mysql_pconnect() is used.

MySQL Connectivity

```
<?php
  $con =
  mysql_connect("localhost",”root",”admin");
  if (!$con){
    die('Could not connect: ' . mysql_error());
  }
  $sql = "CREATE DATABASE my_db";
  if (mysql_query($sql,$con)){
    echo "Database my_db created";
  }
  else{
    echo "Error creating database:" .
  mysql_error();
  }
?>
```

MySQL Connectivity

```
<?php
  $con = mysql_connect("localhost", “root",
                                  “admin");
  if (!$con){
    die('Could not connect: '. mysql_error());
  }
  $db_selected = mysql_select_db("test_db", $con);
  if (!$db_selected){
    die ("Can\'t use test_db : “. mysql_error());
  }
  $sql="SELECT * FROM Person where name=‘$uname’";
  mysql_query($sql,$con);
  mysql_close($con);
?>
```

MySQL Connectivity

- **`mysql_fetch_array()`**
  - The **`mysql_fetch_array()`** function returns a row from a recordset as an associative array and/or a numeric array.
  - This function gets a row from the **`mysql_query()`** function and returns an array on success, or FALSE on failure or when there are no more rows.
- Syntax
  - **`mysql_fetch_array(data,array_type)`**

MySQL Connectivity

Parameter	Description
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function
array_type	Optional. Specifies what kind of array to return.Possible values: •MYSQL_ASSOC - Associative array •MYSQL_NUM - Numeric array •MYSQL_BOTH - Default. Both associative and numeric array

MySQL Connectivity

- **`mysql_fetch_row()`**
  - The **`mysql_fetch_row()`** function returns a row from a recordset as a numeric array.
  - This function gets a row from the **`mysql_query()`** function and returns an array on success, or FALSE on failure or when there are no more rows.
- Syntax

Parameter	Description
data	Required. Specifies which data pointer to use. The data pointer is the result from the mysql_query() function

MySQL Connectivity

```
<?php
error_reporting(E_ALL ^ E_DEPRECATED);
$con = mysql_connect("localhost", "root", "");
if (!$con){ die('Could not connect: '.mysql_error());}
$db_selected = mysql_select_db("test_db", $con);
if (!$db_selected){
        die ("Can\'t use test_db : ". mysql_error());
        }
$uname = "a.rahman";
$sql="SELECT * FROM Person";
$result = mysql_query($sql,$con);

echo "<table border='1' >";
while ($row = mysql_fetch_row($result)) {
        echo "<tr><td>".$row['0']."</td><td>".$row[1].
        "</td><td>".$row[2]."</td><td>".
                        $row[3]."</td></tr>";
}
echo "</table>";
mysql_close($con);
?>
```

localhost/week08/ex05-mysql.php

1	abdul rahman	112323423	352345
2	kamran	5462345345	562456
3	ayesha	2352342345	432452

MySQL Connectivity

```
<?php
  echo "<table>";
  while ($row = mysql_fetch_array($result))
  {
      echo "<tr><td>{$row['ID']}
      </td><td>{$row[1]}
      </td><td>{$row[2]}</td><td>
                  ${row['cellno']}</td></tr>";
  }
  echo "</table>";
?>
```

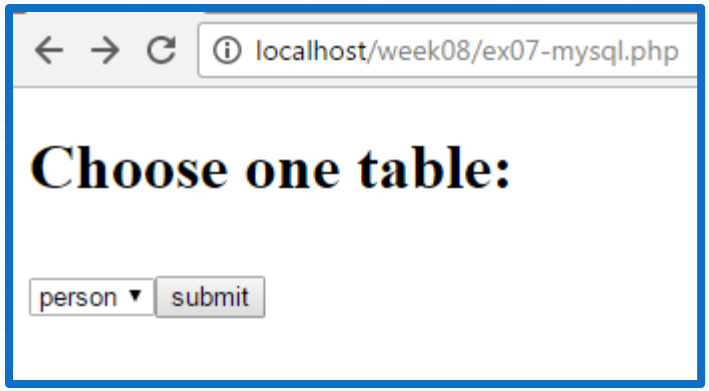
Ex – show data in the tables

- Function: list all tables in your database. Users can select one of tables, and show all contents in this table.

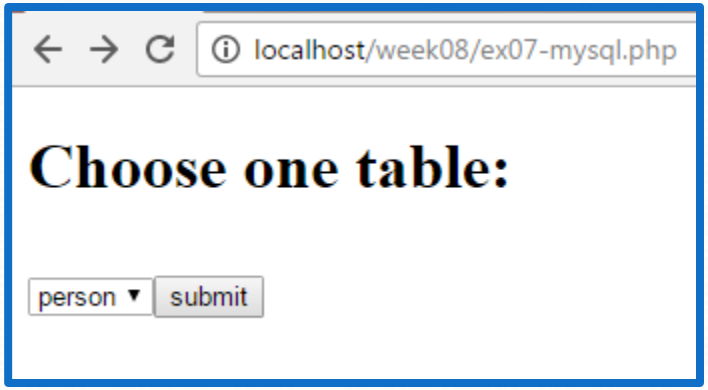
- ex07.php
- showtable.php

ex07.php

```
<html><head><title>MySQL Table Viewer</title></head><body>
<?php
$dbhost = ‘localhost:3306';
$dbuser = ‘root';
$dbpass = ‘’;
$dbname = ‘test_db’;
//$table = ‘person';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if (!$conn) {
    die('Could not connect: ' . mysql_error());
}
if (!mysql_select_db($dbname))
    die("Can't select database");
```



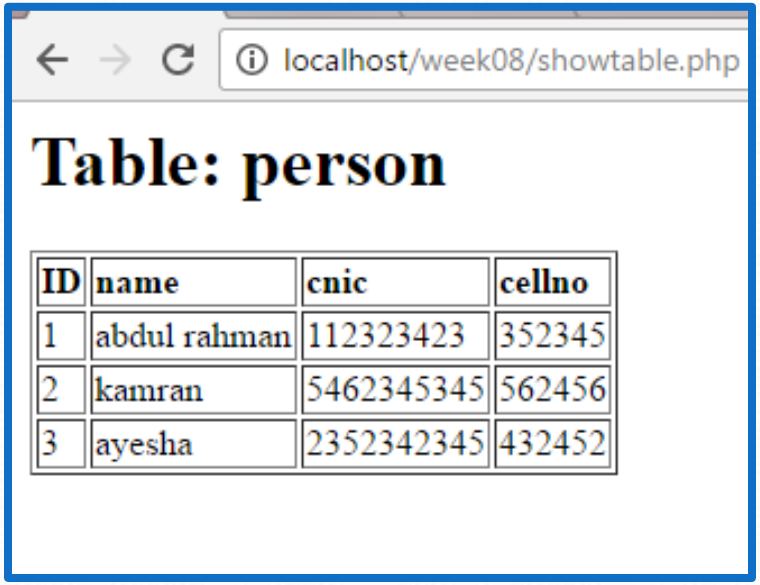
ex07.php




```
$result = mysql_query("SHOW TABLES");
if (!$result) {
    die("Query to show fields from table failed");
}
$num_row = mysql_num_rows($result);
echo "<h1>Choose one table:<h1>";
echo "<form action=\"showtable.php\" method=\"POST\">";
echo "<select name=\"table\" size=\"1\" Font size=\"+2\">";
for($i=0; $i<$num_row; $i++) {
     $tablename=mysql_fetch_row($result);
     echo "<option value=\"{$tablename[0]}\" >{$tablename[0]}</option>";
}
echo "</select>";
echo "<input type=\"submit\" value=\"submit\">";
echo "</form>";

mysql_free_result($result);
mysql_close($conn);
?>
</body></html>
```

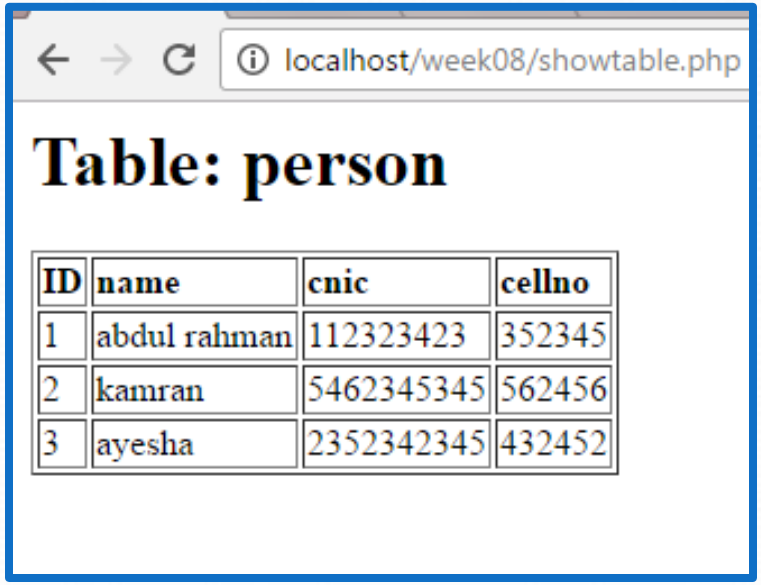
showtable.php




```
<html><head>
<title>MySQL Table Viewer</title>
</head>
<body>
<?php
$dbhost = 'hercules.cs.kent.edu:3306';
$dbuser = 'nruan';
$dbpass = ‘**********’;
$dbname = 'nruan';
$table = $_POST[“table”];
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if (!$conn)
    die('Could not connect: ' . mysql_error());
if (!mysql_select_db($dbname))
    die("Can't select database");
$result = mysql_query("SELECT * FROM {$table}");
if (!$result)  die("Query to show fields from table failed!" . mysql_error());
```

showtable.php (cont.)

```
$fields_num = mysql_num_fields($result);
echo "<h1>Table: {$table}</h1>";
echo "<table border='1'><tr>";
// printing table headers
for($i=0; $i<$fields_num; $i++) {
    $field = mysql_fetch_field($result);
    echo "<td><b>{$field->name}</b></td>";
}
echo "</tr>\n";
while($row = mysql_fetch_row($result)) {
    echo "<tr>";
    // $row is array... foreach( .. ) puts every element
    // of $row to $cell variable
    foreach($row as $cell)
            echo "<td>$cell</td>";
    echo "</tr>\n";
}
mysql_free_result($result);
mysql_close($conn);
?>
</body></html>
```



ID	name	cnic	cellno
1	abdul rahman	112323423	352345
2	kamran	5462345345	562456
3	ayesha	2352342345	432452

Functions Covered

- mysql_connect() mysql_select_db()
- include()
- mysql_query() mysql_num_rows()
- mysql_fetch_array() mysql_close()
- mysql_fetch_field()

Cookies

Benefit of Cookies

- Cookies are used for authenticating, tracking, and maintaining specific information about users
- Personolised home pages
- Electronic shopping carts.

Why use sessions

- A normal HTML website will not pass data from one page to another
- All information is forgotten when a new page is loaded
- Many websites need to pass user data from one page to another
  - for tasks like a shopping cart, which requires data(the user's selected product) to be remembered from one page to the next
- Using PHP sessions is one solution.

Sessions

- The **`session_start()`** function is used to create a session. Should be called before **`<html>`** tag.

```
<?php
session_start();
?>
```

Sessions

```
<?php
  session_start();
  if (!isset($_SESSION['count']))
      $_SESSION['count'] = 0;
  else
       $_SESSION['count']++;
?>
```

Sessions

- **`session_unregister(´varname´);`** unregisters a session variable
- **`session_destroy()`** destroys a session

Cookies

- The **`setcookie()`** function is used to create cookies. Should be called before **`<html>`** tag.
- **`setcookie(name, [value], [expire], [path], [domain], [secure]);`**
- **`<?php setcookie("uname", $name, time()+36000); ?>`**
- This sets a cookie named "**`uname`**" - that expires after ten hours.
- Either a blank value or a time in the past makes the cookie expired.

Cookies

- To access a cookie, refer to the cookie name as a variable or use **$_COOKIE** array. The **isset()** checks whether the cookie is set or not

```
<html> <body>
<?php
  if (isset($uname))// isset($_Cookie[$uname])
      echo "Welcome " . $_Cookie[$uname].
                                  "!<br />";
  else
      echo "You are not logged in!<br />"; ?>
?>
</body> </html>
```

Defining PHP Classes

```
<?php
  class phpClass {
    var $var1;
    var $var2 = "constant string";

    function myfunc ($arg1, $arg2) {
      [..]
    }
    [..]
  }
?>
```

Defining PHP Classes

- The special form **class**, followed by the name of the class that you want to define.
- A set of braces enclosing any number of variable declarations and function definitions.
- Variable declarations start with the special form **var**, which is followed by a conventional $ variable name; they may also have an initial assignment to a constant value.
- Function definitions look much like standalone PHP functions but are local to the class and will be used to set and access object data.

```
class Books {
    /* Member variables */
    var $price;
    var $title;
    /* Member functions */
    function setPrice($par){
      $this->price = $par;
    }
    function getPrice(){
      echo $this->price ."<br/>";
    }
    function setTitle($par){
      $this->title = $par;
    }
    function getTitle(){
      echo $this->title ." <br/>";
    }
  }
```

Creating Objects in PHP

```
$physics = new Books;
$maths = new Books;
$chemistry = new Books;
```

- Here we have created three objects and these objects are independent of each other and they will have their existence separately. Next we will see how to access member function and process member variables.

Calling Member Functions

- To set title & prices for 3 books by calling member functions:

```
$physics->setTitle( "Physics for High School" );
$chemistry->setTitle( "Advanced Chemistry" );
$maths->setTitle( "Algebra" );
$physics->setPrice( 10 );
$chemistry->setPrice( 15 );
$maths->setPrice( 7 );
```

- call another member functions to get the values set by in above example:

```
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();
$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
```

This will produce the following result:
Physics for High School
Advanced Chemistry
Algebra
10
15
7

Constructor Functions

- PHP provides a special function called __**construct()** to define a constructor. You can pass as many as arguments you like into the constructor function.
- Following example will create one constructor for Books class and it will initialize price and title for the book at the time of object creation.

```
function __construct( $par1, $par2 ) {
   $this->title = $par1;
   $this->price = $par2;
}
```

Constructor Functions

```
$physics = new Books( "Physics for High School", 10 );
$maths = new Books ( "Advanced Chemistry", 15 );
$chemistry = new Books ("Algebra", 7 );

/* Get those set values */
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();

$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
```

This will produce the following result :

```
Physics for High School
Advanced Chemistry
Algebra
10
15
7
```

Destructor

- Like a constructor function you can define a destructor function using function __**destruct()**. You can release all the resources with-in a destructor.

Inheritance

```
class Child extends Parent {
   <definition body>
}
```

- The child class (or subclass or derived class) has the following characteristics:
- Automatically has all the member variable declarations of the parent class.
- Automatically has all the same member functions as the parent, which (by default) will work the same way as those functions do in the parent.

Inheritance

- Following example inherit Books class and adds more functionality based on the requirement.

```
class Novel extends Books {
   var $publisher;

   function setPublisher($par){
      $this->publisher = $par;
   }

   function getPublisher(){
      echo $this->publisher. "<br />";
   }
}
```

Function Overriding

- Function definitions in child classes override definitions with same name in parent classes. In a child class, we can modify the definition of a function inherited from parent class.
- In the following example getPrice and getTitle functions are overridden to return some values.

```
function getPrice() {
   echo $this->price . "<br/>";
   return $this->price;
}
```

```
function getTitle(){
   echo $this->title . "<br/>";
   return $this->title;
}
```

Public Members

- Unless you specify otherwise, properties and methods of a class are public. That is to say, they may be accessed in three possible situations −
- From outside the class in which it is declared
- From within the class in which it is declared
- From within another class that implements the class in which it is declared

Private members

- By designating a member private, you limit its accessibility to the class in which it is declared.

```
class MyClass {
  private $car = “secret";
  $driver = “Khan";
  function __construct($par) {
    // Statements here run every time
    // an instance of the class
    // is created.
  }
  function myPublicFunction() {
    return("I'm visible!");
  }
  private function myPrivateFunction() {
    return("I'm  not visible outside!");
  }
}
```

The extending class will not have any awareness of or access to **myPrivateFunction** and **$car,** because they are declared private.

Protected members

- A protected property / method is accessible in the class in which it is declared, in classes that extend that class, but are not available outside.

```
class MyClass {
   protected $car = “secret";
   $driver = “Khan";
   function __construct($par) {
      // Statements here run every time
      // an instance of the class is created
      // }
   function myPublicFunction() {
      return("I'm visible!");
   }
   protected function myPrivateFunction() {
      return("I'm  visible in child class!");
   }
}
```

Abstract Classes

- An abstract class is one that cannot be instantiated, only inherited.

```
abstract class MyAbstractClass {
   abstract function myAbstractFunction() {
   }
}
```

Calling parent constructors

```
class Name {
   var $_firstName;
   var $_lastName;
   function Name($first_name, $last_name) {
      $this->_firstName = $first_name;
      $this->_lastName = $last_name;
   }
   function toString() {
      return($this->_lastName .", " .$this->_firstName);
   }
}
class NameSub1 extends Name {
   var $_middleInitial;
   function NameSub1($first_name, $middle_initial, $last_name) {
      Name::Name($first_name, $last_name);
      $this->_middleInitial = $middle_initial;
   }
   function toString() {
      return(Name::toString() . " " . $this->_middleInitial);
   }
}
```