[IS113] Extra Exercises - Weeks 9 & 10 - Classes & Objects

Objectives

- To master the concepts of classes and objects in PHP

Instructions

- Questions with no asterisk mark are easy peasy.
- Questions marked with * are slightly challenging.
- Questions marked with ** are challenging.
- Questions marked with *** are very challenging.

Download

- **Resources**: Click here
- **Solutions**: Click here

NOTE: If you spot any mistakes/errors in the questions, please contact your instructors by email and state the issues. We will try to address it as soon as possible. *Questions 7-11 are adapted from Java programming class (IS200) which is now defunct.*

Question 1 - Car

Go to **car** directory. Complete the following **Parts A**, **B**, and **C**.

In the last few weeks, we have been using **Associative Arrays** to represent and store “things” such as persons, books, fruits, students, etc. For example,

- **$cars** is a normal **Array** where each car is represented and stored as an **Associative Array**.
    - The **key** is an **attribute** of a car (e.g. year, make, model, rating)

```
$cars = [  [  "year"   => 2008,
            "make"   => 'Honda',
            "model"  => 'Fit',
            "rating" => 7 ],

          [  "year"   => 2016,
            "make"   => 'Hyundai',
            "model"  => 'Sonata',
            "rating" => 6 ],

          [  "year"   => 2000,
            "make"   => 'Toyota',
            "model"  => 'Corolla',
            "rating" => 6 ]        ];
```

Part A (*)

Write a class named **Car** inside **Car.php** file:

- **Car** class defines what a car should be. Every **Car object** has the following FOUR (4) attributes:
    - **year** (e.g. 2002, 2010, 2019, etc.)
    - **make** (e.g. Honda, Hyundai, Kia, Ford, etc.)
    - **model** (e.g. Corolla, Civic, Sonata, Focus, etc.)
    - **rating** (Integer on a scale of 1 to 10)
- Implement its constructor so that it takes in values for the FOUR (4) attributes
- Implement **Getter** methods for all of the attributes.

Part B (*)

Complete **cars.php** file.

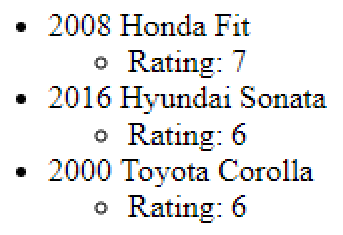
- It uses the **Car class definition** from **Car.php** file.
- Write code to instantiate THREE (3) **new Car objects**.
- Store these **new Car objects** in a normal **Array** with the variable name **$cars**.

Part C (**)

Complete **show.php** file.

- It references **$cars Array** (from **cars.php** file) and displays the **Car objects** as shown below (specific values for the attributes may vary - depending on your own input in Part B):

show.php



- 2008 Honda Fit
    - Rating: 7
- 2016 Hyundai Sonata
    - Rating: 6
- 2000 Toyota Corolla
    - Rating: 6

Question 2 - Drinks

Go to **drinks** directory. Complete the following **Parts A, B**, **C**, and **D**.

Krazy Korean Store plans to sell alcoholic drinks on their brand new website. However, their website is incomplete and broken. You are tasked to fix it as per the following requirements.

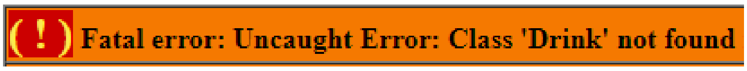
Part A (*)

Complete **Drink.php** file:

- Implement its constructor so that it takes in values for the THREE (3) attributes of **Drink** class;
- Implement **Getter** and **Setter** methods for the THREE (3) attributes of **Drink** class.

Part B (*)

Fix **drinks.php** file so that it is **error-free**. Currently, it displays the following error message in web browser:

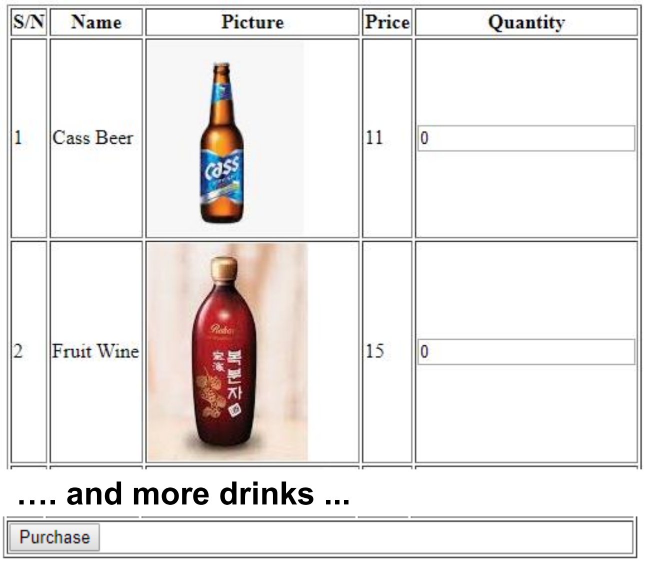


(!) Fatal error: Uncaught Error: Class 'Drink' not found

Part C (**)

Complete **index.php** file. It imports **drinks.php** file, from which it obtains an **Array** of **Drink** objects (**$drinks**). The **index.php** page, when opened in a web browser, must display all drinks and their information as shown below.

index.php



S/N	Name	Picture	Price	Quantity
1	Cass Beer		11	0
2	Fruit Wine		15	0

…. and more drinks ...

- Display all the information in an HTML table as shown above. Use Array **$drinks**.
  - Do NOT hardcode **S/N**. It must be dynamically generated in your code.
  - The displayed images are **thumbnail** images (in **images/thumbnail/** folder).
  - When the user clicks on a **thumbnail** image, display the corresponding large image in a **new browser window/tab**. The large image files are in **images/** folder.
- The user must be able to:
  - Key in **quantity** for each **drink** in a form **INPUT** field;
  - Click on the **Purchase** submit button at the bottom to submit the **form**. It must then take the user to **purchase.php** page.

Part D (***)

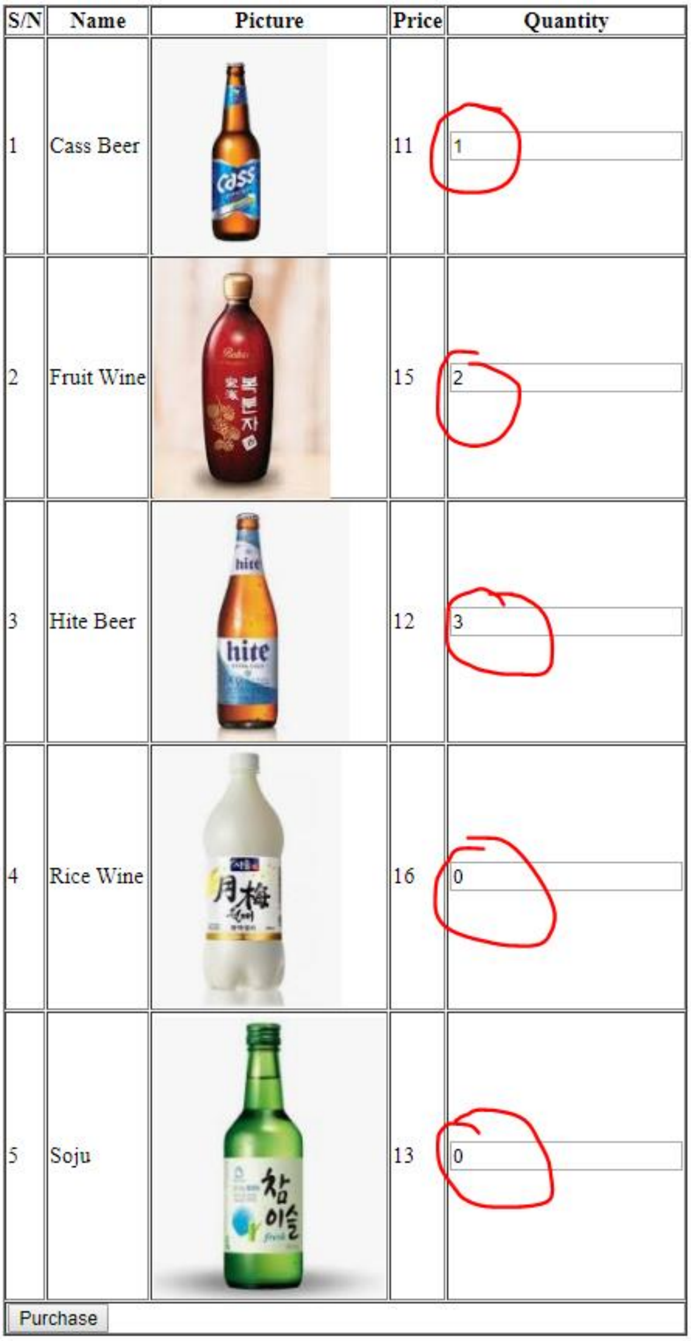
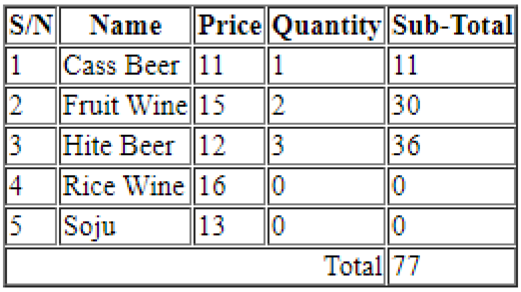
Complete **purchase.php** file. This page:

1. Display all drinks and their information. Use Array **$drinks**.
   a. Do NOT hardcode **S/N**. It must be dynamically generated in your code.
2. Retrieves the user's **form input** (e.g. **quantity**) from **index.php** page.

3. Calculates **sub-total** ($) based on the user-specified **quantity** (for each drink choice) and display it ("Sub-Total" column).
4. Calculates **total** ($) and display it (in "Sub-Total" column and on the right hand side of the text **Total**).

IMPORTANT: Think creatively about how to obtain the **quantity** associated with **each drink choice** in **index.php** page.

Example

index.php	purchase.php
	

S/N	Name	Picture	Price	Quantity
1	Cass Beer		11	1
2	Fruit Wine		15	2
3	Hite Beer		12	3
4	Rice Wine		16	0
5	Soju		13	0

S/N	Name	Price	Quantity	Sub-Total
1	Cass Beer	11	1	11
2	Fruit Wine	15	2	30
3	Hite Beer	12	3	36
4	Rice Wine	16	0	0
5	Soju	13	0	0
			Total	77

Question 3 - Dating

Go to **dating** directory. Complete the following **Parts A, B**, **C**, and **D**.

Krazy Dating plans to launch a new website. The website would allow users to find their dream dates. However, their website is incomplete and broken. You are tasked to fix it as per the following requirements.

Part A (*)

Complete **Person.php** file:

- Implement its constructor so that it takes in values for the SIX (6) attributes of Person class;
- Some of the Getter and Setter methods are **broken**. Fix them.

Part B (**)

Previously in **Question 1 (Drinks)**, we created multiple **Drink** objects in **drinks.php** file. The **Drink** objects were then stored in **$drinks Array**. Importing **drinks.php** file using **require_once** allowed other PHP pages to access this **$drinks Array**.

In **Dating** exercise, we declare a **DAO** (Data Access Object) **class** inside **PersonDAO.php** file. Here is the code snippet of this **DAO**.

PersonDAO.php

```
<?php
require_once 'Person.php';

class PersonDAO {
    private $people;

    // Constructor
    public function __construct() {
        $this->people  = array(
            new Person("Tonald Drump", 'M', 72, 188, 150, 'tonald.jpg'),
            new Person("Rack Bama", 'M', 36, 185, 68, 'rack.jpg'),
            new Person("Rom Cruise", 'M', 23, 168, 58, 'rom.jpg'),
            new Person("Selena Goaway", 'F', 20, 160, 45, 'selena.jpg'),
            new Person("Hailey Bald", 'F', 18, 175, 60, 'hailey.jpg'),
            new Person("Nicole Who", 'F', 35, 180, 58, 'nicole.jpg')
        );
    }

    public function getPeople() {
        return $this->people;
    }
    ... more code below ...
}
```

PersonDAO class's constructor pre-populates **$people Array** with SIX (6) **Person** objects. And, via its **getPeople()** public method, **PersonDAO** class allows other PHP pages to access the **$people Array**.

- **DAO** classes typically provide methods for interaction with **Database** (e.g. select, insert, update, etc.).
    - You will learn more about this and modify the above **DAO** class in **Week 10** in IS113.
- **For now**, we will leverage **DAO** classes for pre-populating data and allowing other PHP pages to access the data.

Complete **PersonDAO.php** file:

- Implement **getPeopleByGender()** public method.
- This method takes ONE (1) parameter, **$gender**, where the valid values are:
    - 'M'
    - 'F'
- Given the parameter value of 'M', it is to:
    - Look for one or more **Person** objects in **$people Array** where each person's **gender** is 'M'
    - Insert all matching **Person** objects into an Array and return that Array.

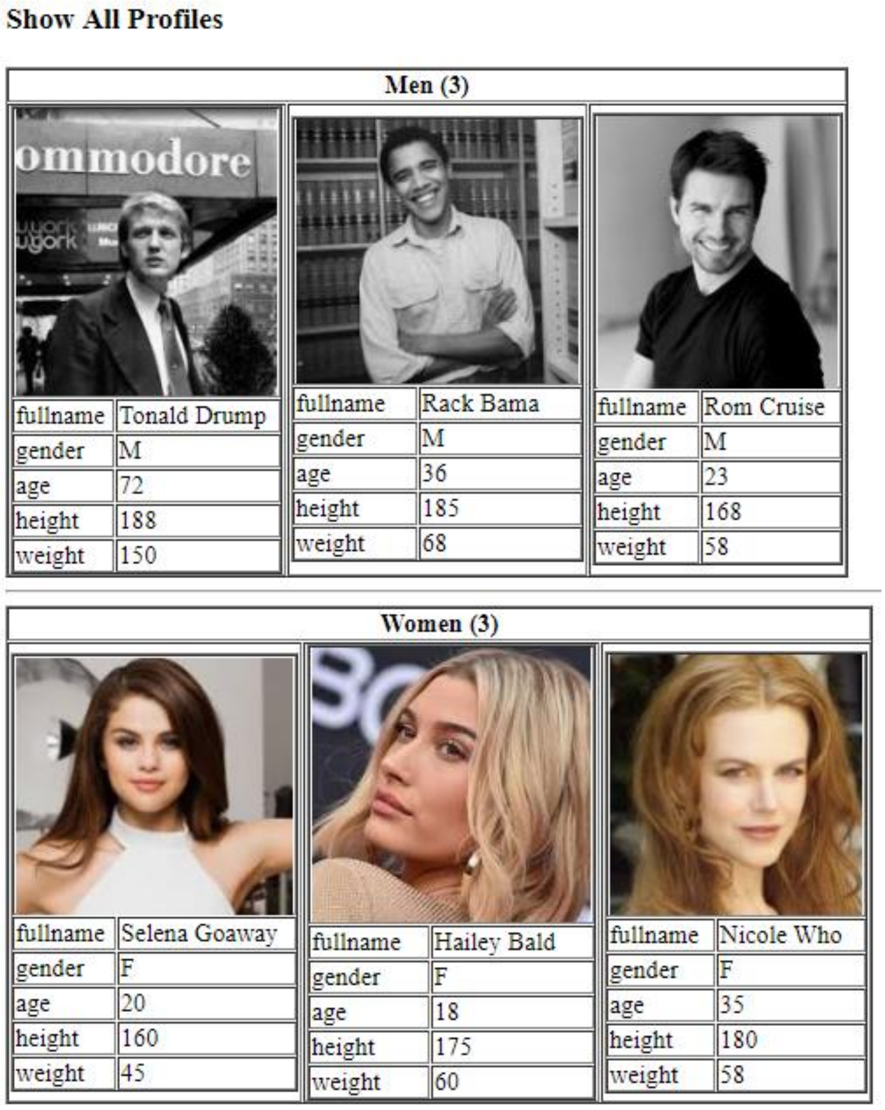
- Likewise, for the parameter value of 'F', it is to perform the same but this time, the returned Array will contain all **Person** objects where each person’s **gender** is 'F'.

Part C (***)

Complete **index.php** file.

- It imports **PersonDAO.php** file, with which it can call all **public methods** of **PersonDAO** class.
- For example, by calling **getPeople()** public method, it is able to obtain an **Array** of **Person** objects.
- It can also call **getPeopleByGender()** public method to obtain an **Array** of **Person** objects of a certain **gender**.
- The **index.php** page, when opened in a web browser, must display all people and their information as shown below.

index.php



Show All Profiles

Men (3)

fullname	Tonald Drump
gender	M
age	72
height	188
weight	150

fullname	Rack Bama
gender	M
age	36
height	185
weight	68

fullname	Rom Cruise
gender	M
age	23
height	168
weight	58

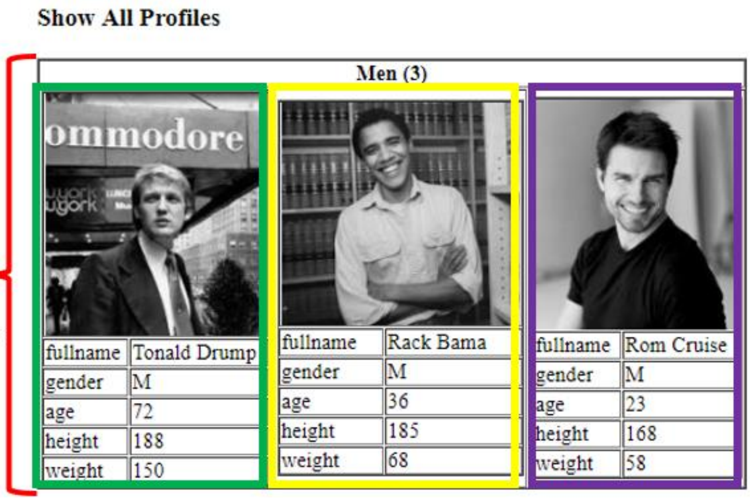
Women (3)

fullname	Selena Goaway
gender	F
age	20
height	160
weight	45

fullname	Hailey Bald
gender	F
age	18
height	175
weight	60

fullname	Nicole Who
gender	F
age	35
height	180
weight	58

- The page displays TWO (2) separate **Tables**.
  - First (top) **Table** displays men.
  - Second (bottom) **Table** displays women.
- First, display all **men** (Person object’s gender value is 'M').
  - Each **Person**’s information is to be displayed in an HTML **Table**. Since there are THREE (3) men, there will be THREE (3) HTML **Tables**.
  - Place all THREE (3) HTML **Tables** in a single row in another HTML **Table**.



Show All Profiles

Men (3)

fullname	Tonald Drump
gender	M
age	72
height	188
weight	150

fullname	Rack Bama
gender	M
age	36
height	185
weight	68

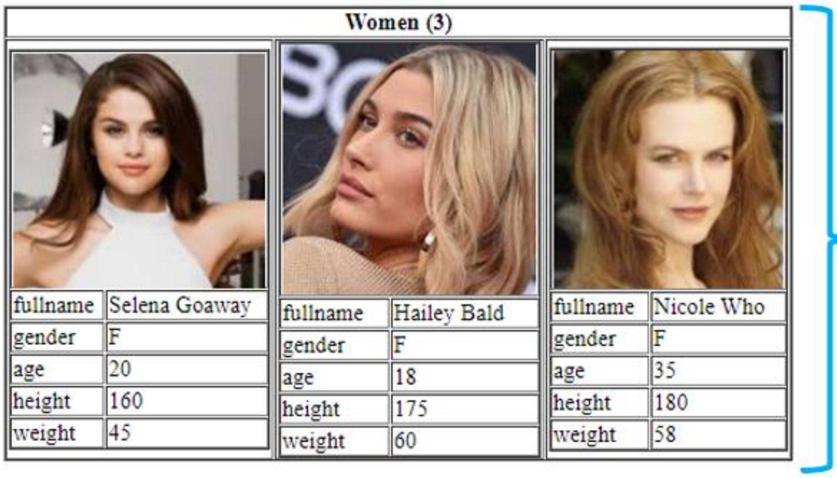
fullname	Rom Cruise
gender	M
age	23
height	168
weight	58

Outer Table #1

- **Consists of 2 rows**
  - **1 Header row**
  - **1 Content row**

- **In the Content row (2nd row):**
  - **Each Person is a new Table**
  - **The 3 tables are placed in a single row**

- Repeat the above step for all **women** (Person object's gender value is 'F').



Women (3)

fullname	Selena Goaway
gender	F
age	20
height	160
weight	45

fullname	Hailey Bald
gender	F
age	18
height	175
weight	60

fullname	Nicole Who
gender	F
age	35
height	180
weight	58

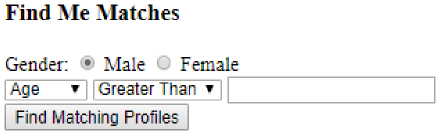
Outer Table #2

- **Consists of 2 rows**
  - **1 Header row**
  - **1 Content row**

Part D (***)

Complete **match.php** file. **match.php** page allows the user to search for desired **dates**. When the page is loaded for the first time, the page looks like this:

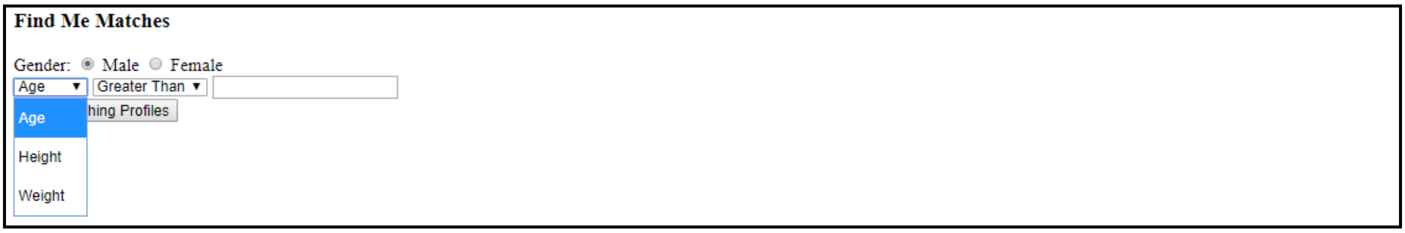
match.php



Find Me Matches

Gender: ● Male ○ Female

The page allows the user to select any of the FOUR (4) attributes: 1) Gender, 2) Age, 3) Height, 4) Weight.

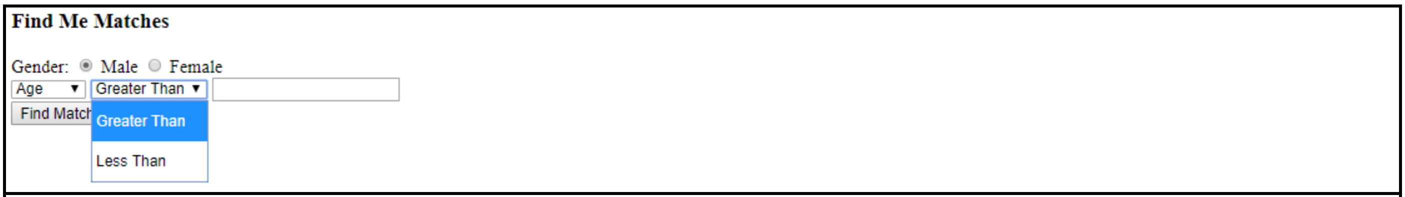


Find Me Matches

Gender: ● Male ○ Female

Age ▼ | Greater Than ▼ |

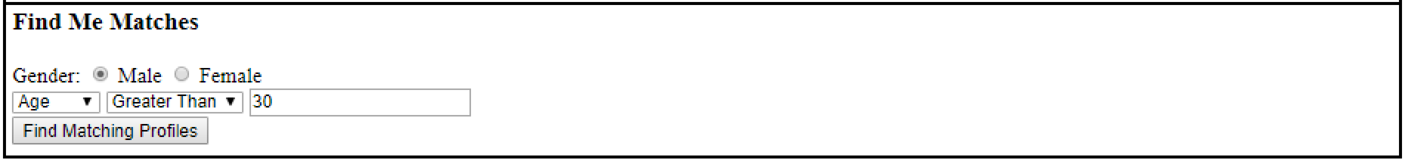
Next, the user can specify the search criteria, e.g. **Male**, **Age** **Greater Than** **30**.



Find Me Matches

Gender: ● Male ○ Female

Age ▼ | Greater Than ▼ |

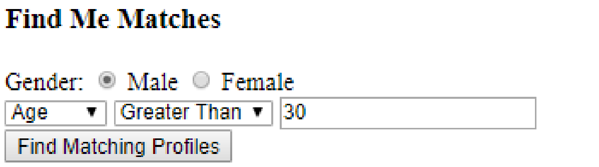
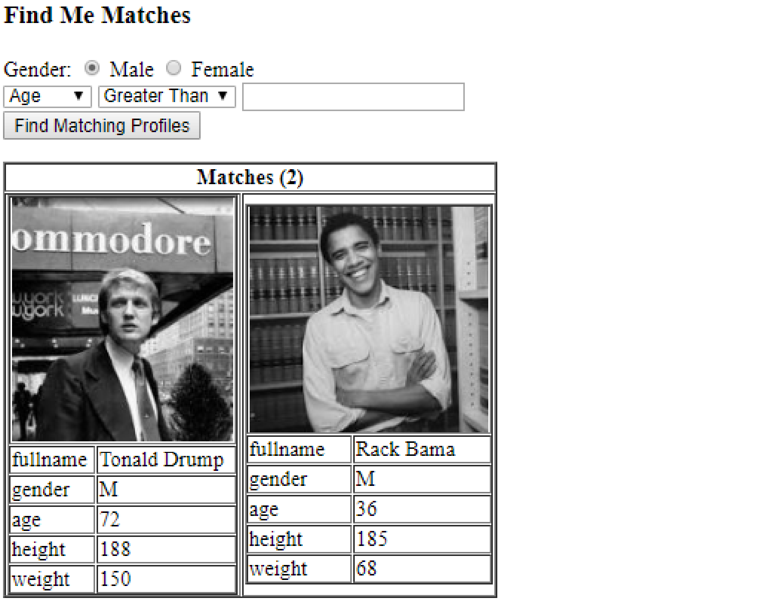


Find Me Matches

Gender: ● Male ○ Female

Age ▼ | Greater Than ▼ | 30

Example

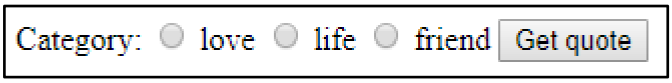
match.php (user input)	match.php (after form submission)
 **Find Me Matches** Gender: ● Male ○ Female Age ▼ Greater Than ▼ 30 Find Matching Profiles	 **Find Me Matches** Gender: ● Male ○ Female Age ▼ Greater Than ▼ Find Matching Profiles Matches (2) fullname: Tonald Drump; gender: M; age: 72; height: 188; weight: 150 fullname: Rack Bama; gender: M; age: 36; height: 185; weight: 68

When one or more **matching** Person objects are found, display them the same way **index.php** displayed Person objects (in an HTML **Table**).

Question 4 - Quotes (**)

Go to **quotes** directory.

1. File index.php contains a form



Category: ○ love ○ life ○ friend

   a. $categories has the list of valid values for the radio buttons.
   b. The form will submit back index.php via HTTP POST

2. WiseMan.php is incomplete.

To do:

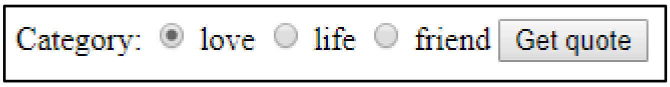
1. Update WiseMan.php according to the comments given in the file.
    a. The number of categories and quotes in the constant QUOTES may be changed in the future, thus, do not hardcode.

2. Update index.php such that
    a. When the page is loaded for the first time, the first category is selected by default.
        i. The number and the order of the elements in $categories may be changed in the future, thus, do not hardcode.

    b. Upon form submission,
        i. Retrieve the user's selected category.
        ii. Instantiate a.k.a create a new WiseMan object.
        iii. Get a random quote using the WiseMan object and assign to $quote.
        iv. Re-populate the form's radio button with user's selection.

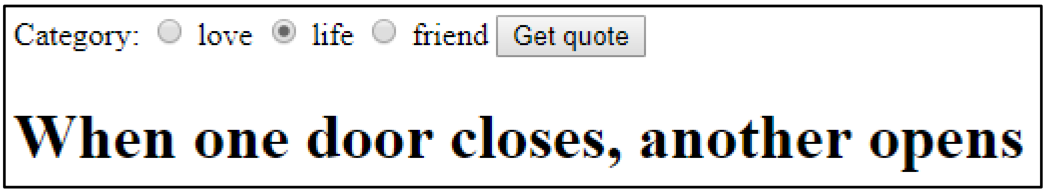
When done correctly,

1. When you load page index.php for the first time,



Category: ● love ○ life ○ friend

2. User selects 'life' and submits. A random quote is displayed.



Category: ○ love ● life ○ friend

When one door closes, another opens

Question 5 - Basket (**)

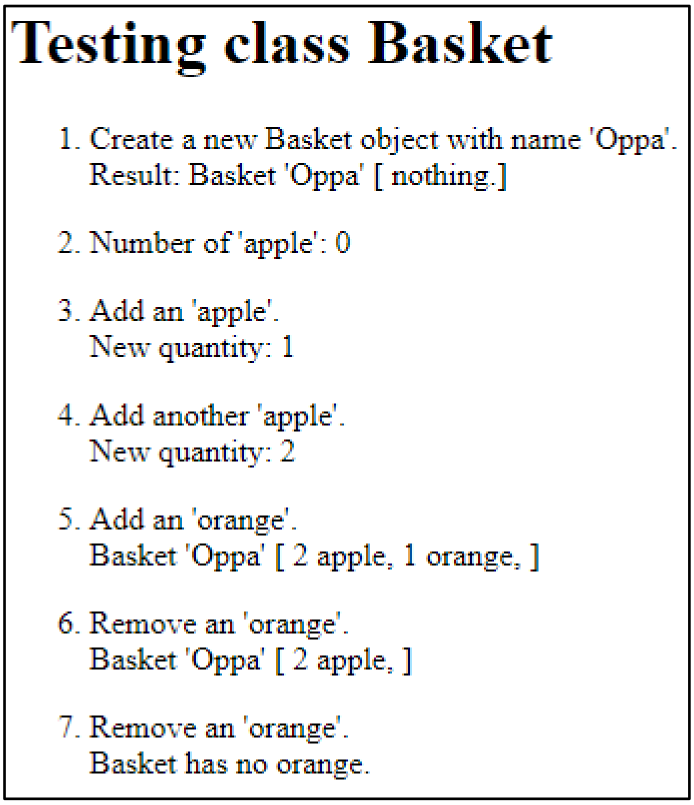
Go to **basket** directory.

1. Basket.php
    a. Class Basket has some missing methods.
    b. Do NOT change the existing code.
    c. Do NOT add attributes to this class.

2. TestBasket.php
    a. Do NOT change this file. This file is correct.
    b. TestBasket.php can’t work now because class Basket has missing methods.
    c. Read and understand the code in TestBasket.php.

To do: Add the missing methods in class Basket based on what is required in TestBasket.php.

If done correctly, the expected output of TestBasket.php is as follows:

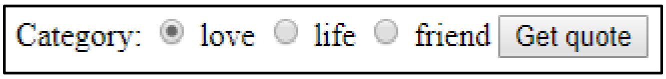


```
Testing class Basket

  1. Create a new Basket object with name 'Oppa'.
     Result: Basket 'Oppa' [ nothing.]

  2. Number of 'apple': 0

  3. Add an 'apple'.
     New quantity: 1

  4. Add another 'apple'.
     New quantity: 2

  5. Add an 'orange'.
     Basket 'Oppa' [ 2 apple, 1 orange, ]

  6. Remove an 'orange'.
     Basket 'Oppa' [ 2 apple, ]

  7. Remove an 'orange'.
     Basket has no orange.
```

Question 6 - Quotes & Likes (**)

Create **quotes_likes** directory, and copy your **quotes** (Question 4) solution here.

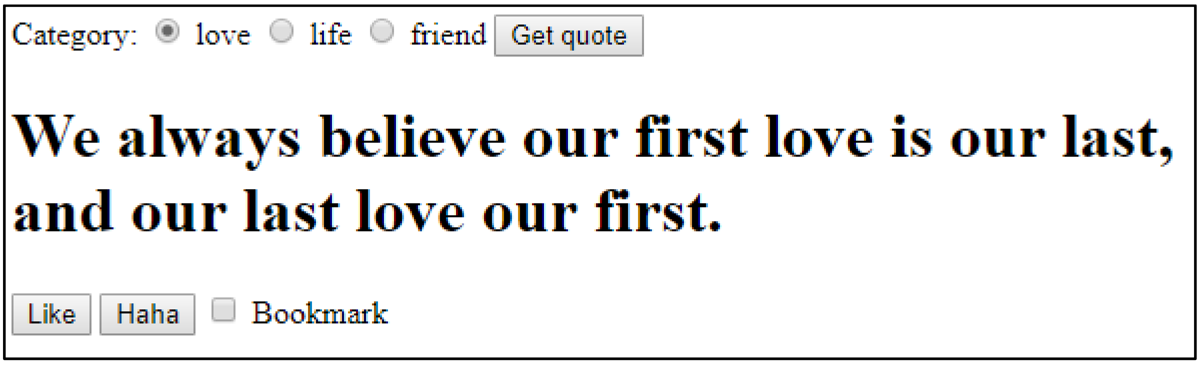
The page index.php has a form that submits to itself via HTTP POST



Category: ● love ○ life ○ friend Get quote

Change the page such that, when the page loads after user clicks ‘Get quote’ button, it displays

- A quote like before, and
- ‘Like’ button, ‘Haha’ button and ‘Bookmark’ checkbox.

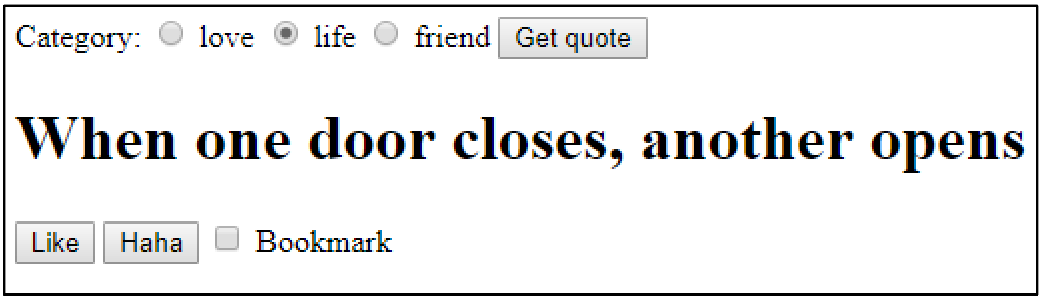


Category: ● love ○ life ○ friend Get quote

We always believe our first love is our last, and our last love our first.

Like Haha □ Bookmark

If user clicks ‘Get Quote’ button again, it displays a new quote, the 2 buttons & checkbox,



Category: ○ love ● life ○ friend Get quote

When one door closes, another opens

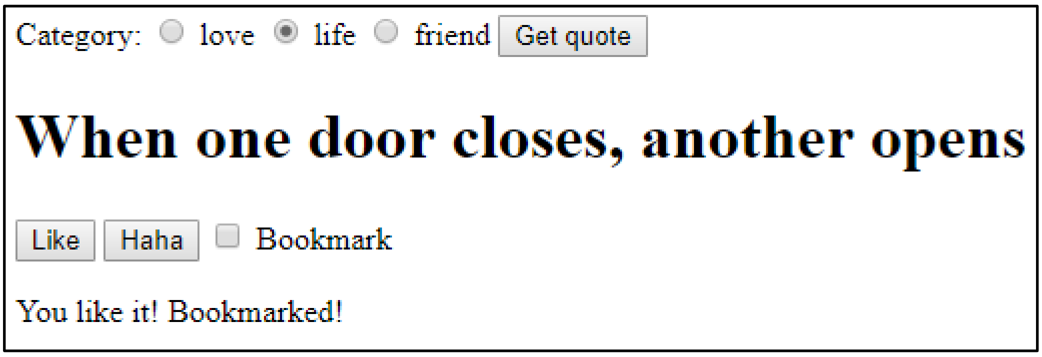
Like Haha □ Bookmark

Ref: ‘[IS113] Extra Exercises - Week 4 - Form Processing > Q6 Like‘

If the user clicks ‘Like’ or ‘Haha’ button,

1. Submits back to index.php via HTTP POST
2. The **same category** is selected.
3. Displays the **same quote** (Hint: https://www.w3schools.com/tags/att_input_type_hidden.asp)
4. Displays a message
   a. For ‘Like’ button, message is ‘You like it!’.
   b. For ‘Haha’ button, message ‘You find it funny.’
   c. ‘Bookmark’ checkbox is ticked, append message with ‘ Bookmarked!’.

Example 1: User ticks 'Bookmark' then clicks 'Like' button.



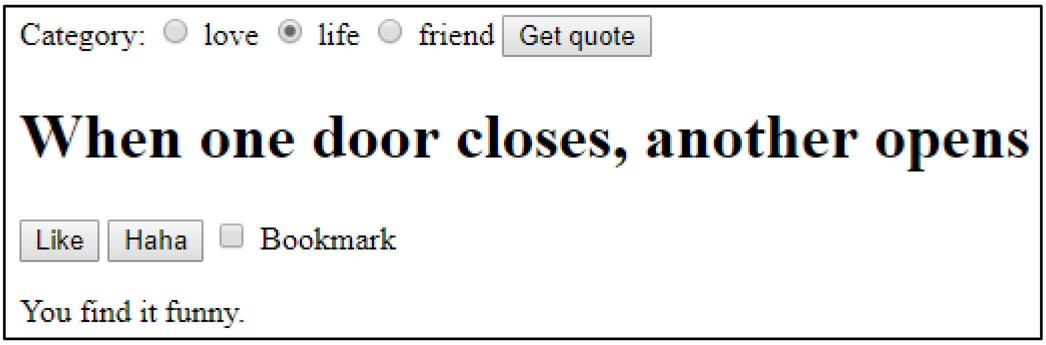
Category: ○ love ● life ○ friend Get quote

When one door closes, another opens

Like Haha □ Bookmark

You like it! Bookmarked!

Example 2: 'Bookmark' is unchecked and user clicks 'Haha' button.



Category: ○ love ● life ○ friend

When one door closes, another opens

□ Bookmark

You find it funny.

Question 7 - Rectangle

Go to **rectangle** directory.

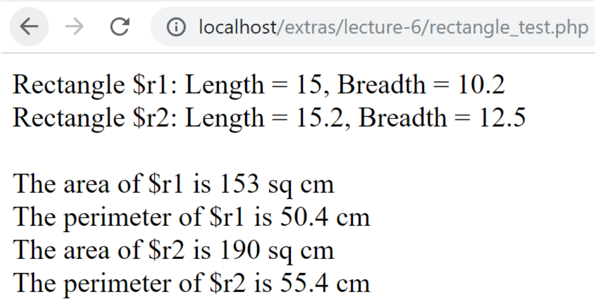
Write a class Rectangle in Rectangle.php with properties length and breadth. The class should contain the following:

i. constructor that configures a new Rectangle with specified length and breadth
ii. getLength()   : returns the length of the Rectangle
iii. getBreadth() : returns the breadth of the Rectangle
iv. setLength($l) : sets the length of the Rectangle to l
v. setBreadth($b): sets the breadth of the Rectangle to b
vi. getArea(): returns the area of the Rectangle calculated as length * breadth
vii. getPerimeter() : returns the perimeter of the Rectangle calculated as 2 * (length + breadth)
viii. toString() : returns a String representing the associated Rectangle, e.g., "Length = 15, Breadth = 10.2"

Complete rectangle_test.php that helps to test the above class. Do the following:

i. create an instance of Rectangle with length 15 cm and breadth 10.2 cm
ii. create another instance of Rectangle of length 15.2 cm and breadth 12.5 cm
iii. use Rectangle's toString() method to print out the length and breadth of the rectangle objects created
iv. Use method getArea and getPerimeter to print out the area and perimeter of the created rectangle objects

rectangle_test.php should display the following when run in your web browser:



```
Rectangle $r1: Length = 15, Breadth = 10.2
Rectangle $r2: Length = 15.2, Breadth = 12.5

The area of $r1 is 153 sq cm
The perimeter of $r1 is 50.4 cm
The area of $r2 is 190 sq cm
The perimeter of $r2 is 55.4 cm
```

Question 8 - Cylinder

Go to **cylinder** directory.

Implement a class Cylinder in Cylinder.php with attributes radius and height. Define PI as a class constant value of 3.14. The class should contain the following:

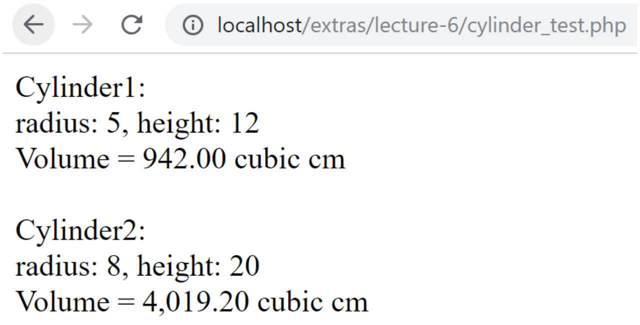
i. Constructor that configures a new cylinder with radius r and height h
ii. getRadius() : returns the radius of the associated cylinder
iii. getHeight() : returns the height of the associated cylinder
iv. setRadius($r): sets the radius of the associated cylinder to r
v. setHeight($h): sets the height of the associated cylinder to h
vi. getVolume(): returns the volume of the cylinder calculated using the formula PI * radius * radius * height
vii. toString(): returns a textual representation of the associated cylinder, e.g., "radius = 5, height = 12"

Write cylinder_test.php that helps to test the above class. Do the following:

i. Create 2 instances of Cylinder with attribute values as

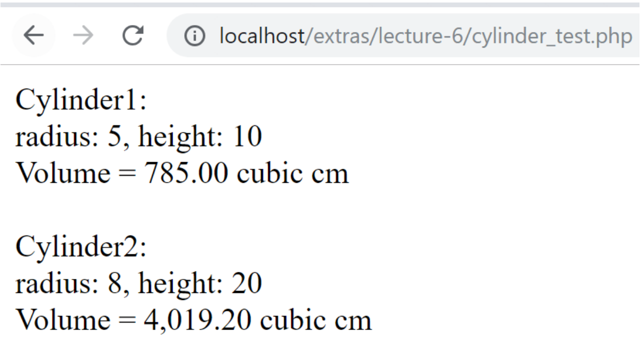
	radius (cm)	height (cm)
cylinder1	5	12
cylinder2	8	20

ii. Print out the attributes and volume of the 2 cylinders created on the console using appropriate methods. Format the volume to 2 decimal places (hint: use number_format built-in function). The following output is observed:



```
Cylinder1:
radius: 5, height: 12
Volume = 942.00 cubic cm

Cylinder2:
radius: 8, height: 20
Volume = 4,019.20 cubic cm
```

iii. Change the height of cylinder1 from 12 to 10 using its setter method. Output the attributes of cylinder1 to the console after the change.

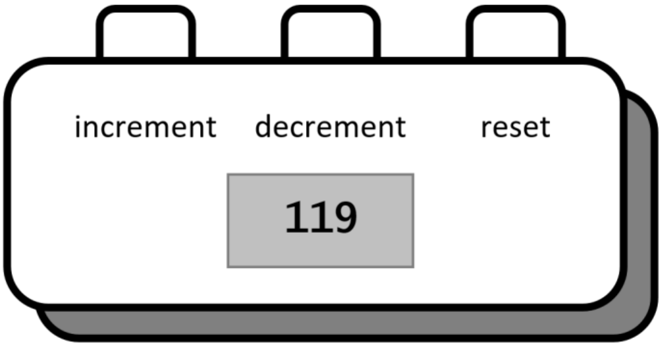


```
Cylinder1:
radius: 5, height: 10
Volume = 785.00 cubic cm

Cylinder2:
radius: 8, height: 20
Volume = 4,019.20 cubic cm
```

Question 9 - Counter (*)

Go to **counter** directory.

Your little sister has a problem with counting. Build a counter that will help her to count. You remember that you used to own a counter that looked something like this:



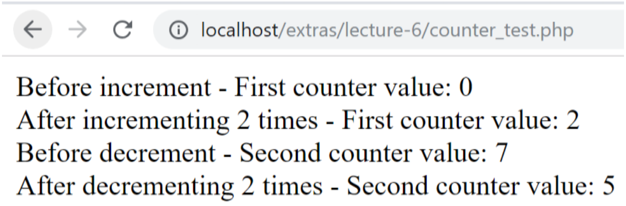
It has an increment button, a decrement button and a reset button that resets the value of the counter to zero.

The class Counter should provide the following:

i. attribute value: that keeps track of the counter value
ii. constructor: that takes in one parameter v and initializes the value to v
iii. getValue(): to get the value of the counter
iv. setValue($newValue): to set the value of the counter to newValue
v. increment(): to increment the value of the counter by 1
vi. decrement(): to decrement the value of the counter by 1
vii. reset(): to set the value of the counter to 0

Complete counter_test.php. Create two instances of Counter class named first and second. Use methods increment() and decrement() to increment and decrement the counter few times and print the values

Sample runs of the program are shown below:



```
Before increment - First counter value: 0
After incrementing 2 times - First counter value: 2
Before decrement - Second counter value: 7
After decrementing 2 times - Second counter value: 5
```

Question 10 - Person

Go to **person** directory.

Implement a class called Person in Person.php. The Person class should have attributes firstName, lastName and age. It should provide the following:

i. constructor to initialize the variables firstName, lastName and age
ii. getFirstName: to return the first name of the person
iii. getLastName: to return the last name of the person
iv. getAge: to return the age of the person
v. setAge: to set the age of the person
vi. isOlder($anotherPerson):return true if this person is older than $anotherPerson
vii. toString(): to return the textual representation of the Person in the format:
Person[name=<firstname> <lastname>, age=<value>]

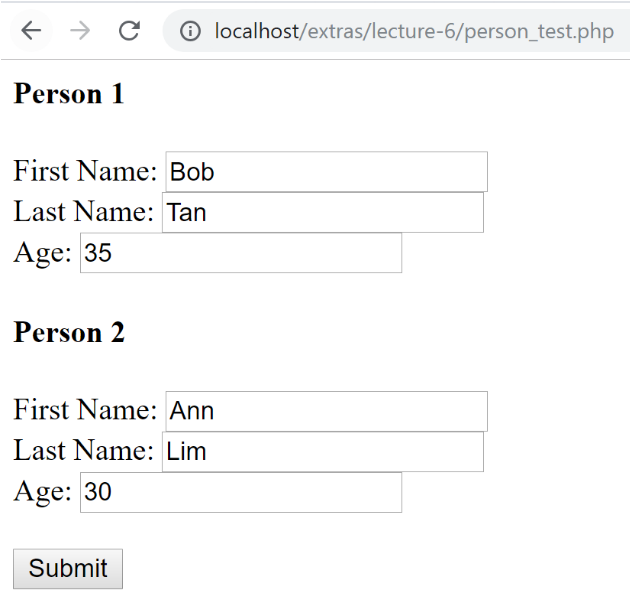
Write person_test.php to test the Person class. It should do the following:

i. Prompt the user to enter first name, last name and age of two particular persons using a HTML form (following self-calling PHP).
ii. Create an instance of both Person objects (say aPerson and bPerson) with values input by the user
iii. Output the details of the older Person object using isOlder and toString methods

Sample runs of person_test.php are shown below:

[Run 1]

person_test.php, Before "Submit" button is clicked:



localhost/extras/lecture-6/person_test.php

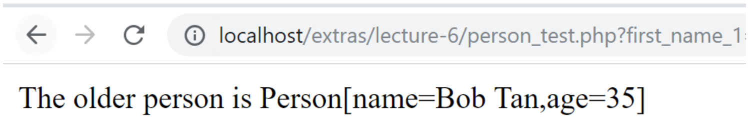
Person 1

First Name: Bob
Last Name: Tan
Age: 35

Person 2

First Name: Ann
Last Name: Lim
Age: 30

person_test.php, After "Submit" button is clicked:

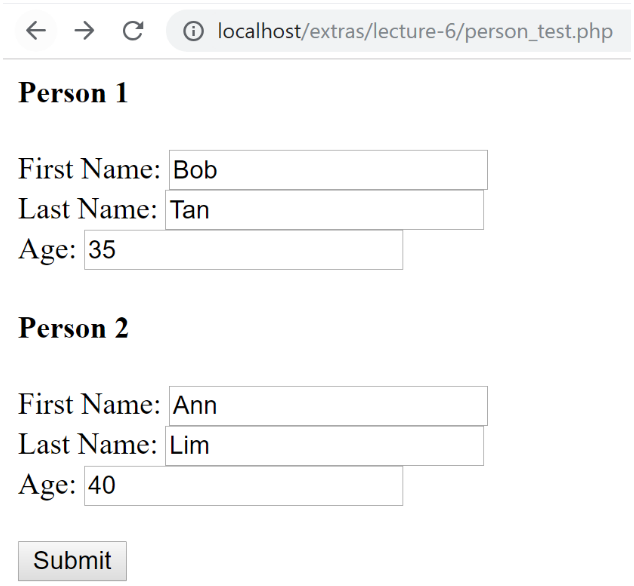


localhost/extras/lecture-6/person_test.php?first_name_1

The older person is Person[name=Bob Tan,age=35]

[Run 2]

person_test.php, Before "Submit" button is clicked:



localhost/extras/lecture-6/person_test.php

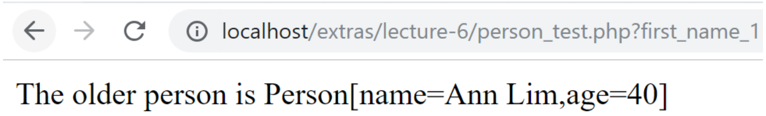
Person 1

First Name: Bob
Last Name: Tan
Age: 35

Person 2

First Name: Ann
Last Name: Lim
Age: 40

person_test.php, After "Submit" button is clicked:

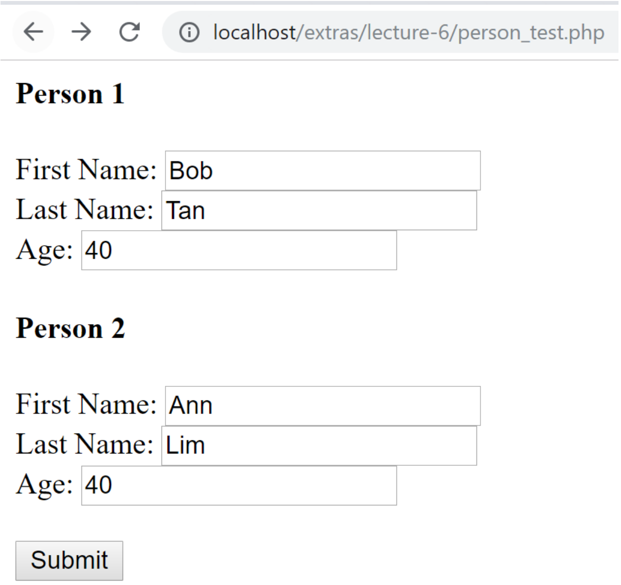


localhost/extras/lecture-6/person_test.php?first_name_1

The older person is Person[name=Ann Lim,age=40]

[Run 3]

person_test.php, Before "Submit" button is clicked:



localhost/extras/lecture-6/person_test.php

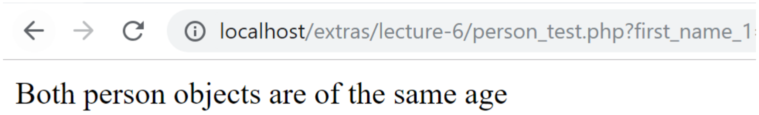
Person 1

First Name: Bob
Last Name: Tan
Age: 40

Person 2

First Name: Ann
Last Name: Lim
Age: 40

person_test.php, After "Submit" button is clicked:



localhost/extras/lecture-6/person_test.php?first_name_1

Both person objects are of the same age

Question 11 - Rational

Go to **rational** directory.

Develop and implement a class called Rational as a representation for manipulating rational numbers. A rational number is the ratio of 2 integers and is represented typically as *a / b* where *a* is the numerator and *b* is the denominator. We assume the denominator to be non-zero.

The basic operations using Rational numbers are performed as follows:

- Addition : $\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$
- Subtraction: $\frac{a}{b} - \frac{c}{d} = \frac{ad - bc}{bd}$
- Multiplication : $\frac{a}{b} \times \frac{c}{d} = \frac{ac}{bd}$
- Division : $\frac{a}{b} \div \frac{c}{d} = \frac{ad}{bc}$

To represent a rational number, you need to represent its numerator and denominator. This necessity implies that a class representing rational numbers needs at least two instance variables – one variable to represent the numerator and the other variable to represent the denominator of the particular object.

```
private $numerator; // Numerator of the associated rational number
 private $denominator; // Denominator of the associated rational number
```

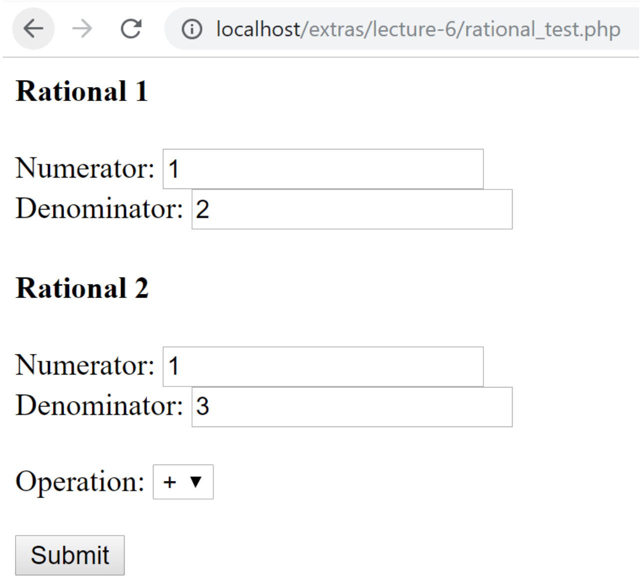
The Rational class should provide the following methods to initialize and manipulate a rational number object in at least the following ways:

- Constructs rational number from a specified numerator and denominator.
- Provides methods for reading and setting the values of numerator and denominator
- Provides methods for adding, subtracting, multiplying, and dividing rational numbers.
- Produces a String representation of a rational number

Implement a client program rational_test.php that can realize the following functionality (note: minor look-and-feel difference is acceptable):

[Run 1]

rational_test.php, Before "Submit" button is clicked:



localhost/extras/lecture-6/rational_test.php

Rational 1

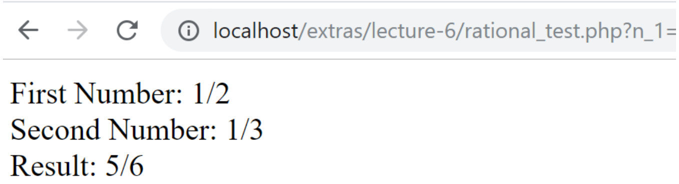
Numerator: 1
Denominator: 2

Rational 2

Numerator: 1
Denominator: 3

Operation: + ▼

rational_test.php, After"Submit" button is clicked:

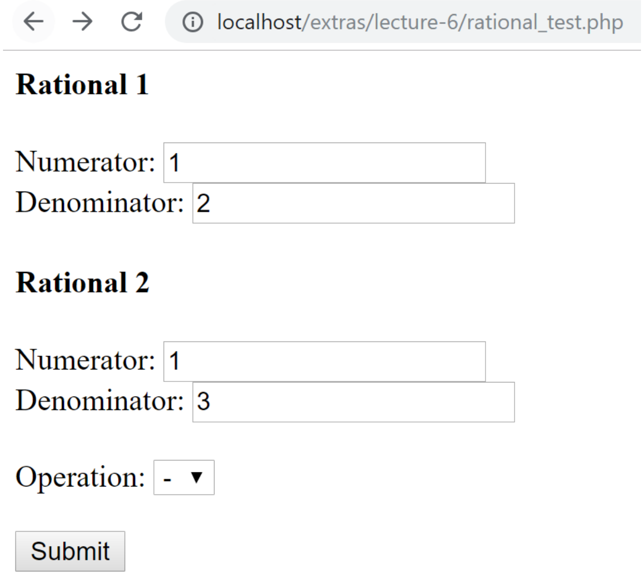


localhost/extras/lecture-6/rational_test.php?n_1=

```
First Number: 1/2
Second Number: 1/3
Result: 5/6
```

[Run 2]

rational_test.php, Before "Submit" button is clicked:



localhost/extras/lecture-6/rational_test.php

Rational 1

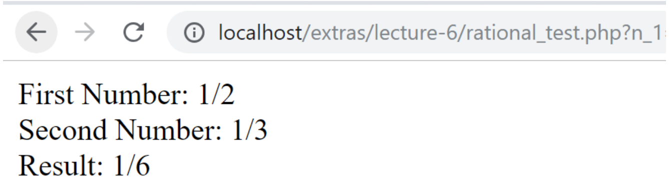
Numerator: 1
Denominator: 2

Rational 2

Numerator: 1
Denominator: 3

Operation: - ▼

rational_test.php, After "Submit" button is clicked:

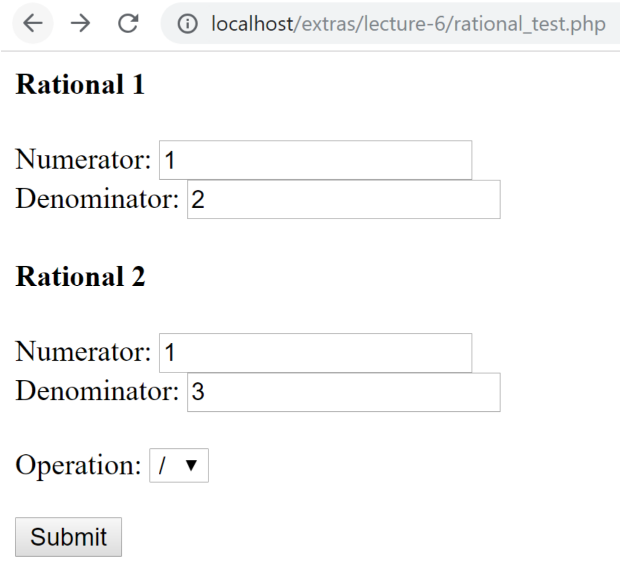


localhost/extras/lecture-6/rational_test.php?n_1

First Number: 1/2
Second Number: 1/3
Result: 1/6

[Run 3]

rational_test.php, Before "Submit" button is clicked:



localhost/extras/lecture-6/rational_test.php

Rational 1

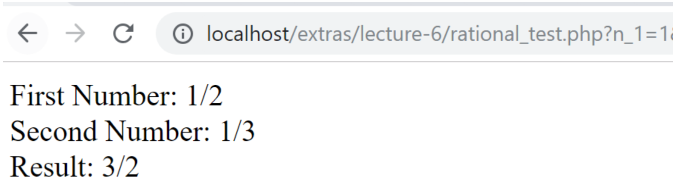
Numerator: 1
Denominator: 2

Rational 2

Numerator: 1
Denominator: 3

Operation: / ▼

rational_test.php, After "Submit" button is clicked:

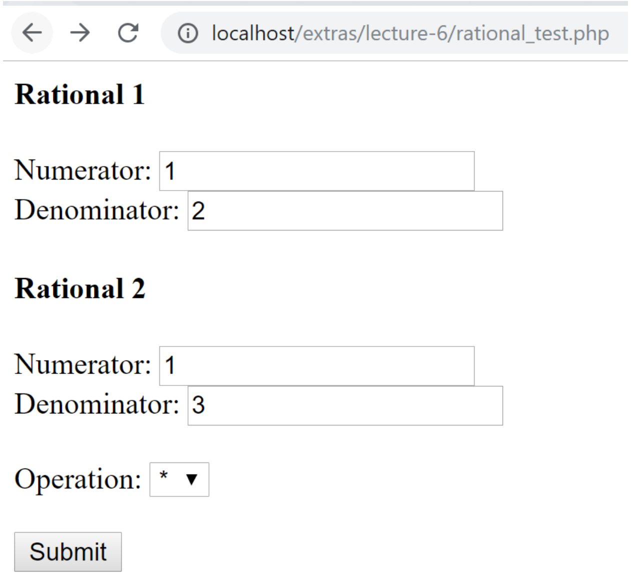


localhost/extras/lecture-6/rational_test.php?n_1=1

First Number: 1/2
Second Number: 1/3
Result: 3/2

[Run 4]

rational_test.php, Before "Submit" button is clicked:



localhost/extras/lecture-6/rational_test.php

Rational 1

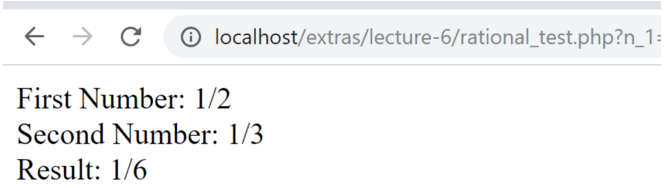
Numerator: 1
Denominator: 2

Rational 2

Numerator: 1
Denominator: 3

Operation: *

rational_test.php, After "Submit" button is clicked:



localhost/extras/lecture-6/rational_test.php?n_1=

```
First Number: 1/2
Second Number: 1/3
Result: 1/6
```

Note: You can assume that all inputs are valid