How to declare classes and methods as abstract?

Step1: **abstract class Car** { }
Step 2: *// Abstract classes are declared with the abstract keyword, // and contain abstract methods*

```
abstract class Car {
abstract public function calcNumMilesOnFullTank(); }
```

Having non abstract methods inside an abstract class

```
abstract class Car {
// Abstract classes can have propertiesprotected $tankVolume;
// Abstract classes can have non abstract methods
public function setTankVolume($volume) {
      $this -> tankVolume = $volume;
   }
 // Abstract method
abstract public function calcNumMilesOnFullTank(); }
```

How to create child classes from an abstract class?

```
class Honda extends Car {
// Since we inherited abstract method,
//    we need to define it in the child class,
//  by adding code to the method's body.
public function calcNumMilesOnFullTank() {
      $miles = $this -> tankVolume*30;

      return $miles;
}
```

Take another class Toyota
Define abstract method calcNumMilesOnFullTank() with a slight change in the calculation
Add its own method with the name of getColor() that returns the string "beige".

```
class Toyota extends Car {
// Since we inherited abstract method,

//    we need to define it in the child class,

//  by adding code to the method's body.public function calcNumMilesOnFullTank() {
return $miles = $this -> tankVolume*33; }
public function getColor() {
return "beige"; }
}
```

Exercise: Let's create a new object, $toyota1, with a full tank volume of 10 Gallons, and make it return the number of miles on full tank as well as the car's color.

Result : ??

Quiz

1 Which keyword is used to declare a class or method as abstract?

A : abstract
B : inherit
C : abstract
D : extends

2 Which keyword is used to declare a child class that inherits from an abstract class?

A : abstract
B : inherit
C : abstract
D : extends

3 What are the main reasons for using an abstract class in the code?

A : To enable code inheritance in PHP.
B : To commit the child classes to certain methods.
C : To make a good impression on your boss and colleagues.
D : To encapsulate the code of the parent from any code outside the parent class.

4 Can we create objects from abstract classes?

Coding exercise

In the following example, we will create an abstract User class and two child classes (Admin andViewer classes) that inherit from the abstract class.

5 Create an abstract class with the name of User, which has an abstract method with the name of stateYourRole.

1. Add to the class a protected property with the name of $username, and public setter and getter methods to set and get the $username.
2. Create an Admin class that inherits the abstract User class.
3. Which method should be defined in the class?
4. Define the method stateYourRole() in the child class and make it return the string “admin”;
5. Create another class, Viewer that inherits the User abstract class. Define the method that should be defined in each child class of the User class.
6. Create an object from the Admin class, set the username to “Balthazar”, and make it return the string “admin”.

How to declare and implement an interface?

We declare an interface with the interface keyword and, the class that inherits from an interface with the implements keyword. Let’s see the general case:

```
interface interfaceName {// Abstract methods
}
class Child implements interfaceName {// Defines the interface methods, // and may have its own methods.
```

```
}
```

In the simple example given below, we will create an interface for the classes that handle cars, which commits all its child classes to setModel() and getModel() methods.

```
interface Car {
public function setModel($name);
public function getModel();
}

class miniCar implements Car {
private $model;
public function setModel($name) {
      $this -> model = $name;
   }

public function getModel() {
return $this -> model; }
}
```

We implement more than one interface in the same class?

```
interface Vehicle {
public function setHasWheels($bool);
public function getHasWheels();
}
// The class implements two interfaces

class miniCar implements Car, Vehicle {private $model;
private $hasWheels;
public function setModel($name) {
      $this -> model = $name;
   }

public function getModel() {
return $this -> model; }
public function setHasWheels($bool) {
      $this -> hasWheels = $bool;
   }

public function getHasWheels() {
return ($this -> hasWheels)? "has wheels" : "no wheels"; }
}
```

Which keyword should we use in order to implement an interface?

A : extends

B : public

C : private
D : implements

Which of the following code types are allowed in an interface?

A : methods
B : properties
C : constants
D : A+C

Which of the following access modifiers is allowed for abstract methods in interfaces?

A : public
B : protected
C : private
D : A+B+C

Which of the following is valid inheritance?

A : A class can inherit from an abstract class and two interfaces.
B : A class can inherit from a parent class and one interface.
C : A class can inherit from a parent class and more than one interface
D : A class can inherit from 2 abstract classes.
E : A+B+C

Coding exercise

1.Create a User class with a protected $username property and methods that can set and get the $username.

2.Create an Author interface with the following abstract methods that can give the user an array of authorship privileges. The first method is setAuthorPrivileges(), and it gets a parameter of $array, and the second method is getAuthorPrivileges().

3.Create an Editor interface with methods to set and get the editor's privileges.

4.Now, create an AuthorEditor class that extends the User class, and implements both the Author and the Editor interfaces.

5. Create in the AuthorEditor class the methods that it should implement, and the properties that these methods force us to add to the class.

For example, in order to implement the public method setAuthorPrivileges(), we must add to our class a property that holds the array of authorship privileges, and name it $authorPrivilegesArray accordingly.

6. Create an object with the name of $user1 from the class AuthorEditor, and set its username to "Balthazar".

7. Set in the $user1 object an array of authorship privileges, with the following privileges: "write text", "add punctuation".

8. Set in the $user1 object an array with the following editorial privileges: "edit text", "edit punctuation".

9. Use the following code to get the $user1 name and privileges:

```
$userName = $user1 -> getUsername();
$userPrivileges = array_merge($user1 -> getAuthorPrivileges(),

  $user1 -> getEditorPrivileges());
```

```
echo $userName . " has the following privileges: ";foreach($userPrivileges as $privilege)
{
        echo " {$privilege},";
 }
echo ".";
```

Implement the polymorphism principle

```
interface Shape {
        public function calcArea();
}
```

Create class Circle implementing the interface -

```
class Circle implements Shape {private $radius;
public function __construct($radius) {
         $this -> radius = $radius;
   }

  // The method calcArea calculates the area of circles

public function calcArea() {
        return $this -> radius * $this -> radius * pi(); }
}
```

Create class Rectangle

```
class Rectangle implements Shape {private $width;
        private $height;

public function __construct($width, $height) {
        $this -> width = $width;

        $this -> height = $height;
   }

  // calcArea calculates the area of rectangles

        public function calcArea() {
                return $this -> width * $this -> height;
        }
}
```

Now create instance

```
$circ = new Circle(3);
$rect = new Rectangle(3,4);

echo $circ -> calcArea();
echo $rect -> calcArea();
```

Result:??

Code Example

```
abstract class User {
protected $scores = 0;protected $numberOfArticles = 0;
  // The abstract and concrete methods

}
```

1 Add to the class concrete methods to set and get the number of articles:

1. setNumberOfArticles($int)
2. getNumberOfArticles() $int stands for an integer.

2. Add to the class the abstract method: calcScores(), that performs the scores calcula- tions separately for each class.

3. Create an Author class that inherits from the User class. In the Author create a concrete calcScores() method that returns the number of scores from the following calculation:numberOfArticles * 10 + 20

4. Also create an Editor class that inherits from the User class. In the Editor create a concrete calcScores() method that returns the number of scores from the following calculation:

numberOfArticles * 6 + 15

5. Create an object, $author1, from the Author class, set the number of articles to 8, and echo the scores that the author gained.

6. Create another object, $editor1, from the Editor class, set the number of articles to 15, and echo the scores that the editor gained.

Code Exercise

1. Create a User interface with set and get methods for both a $username property, as well as for a $gender property.

2. Now, create a Commentator class to implement the User interface.

3. Create function to add “Mr.” or “Mrs.” to the username. When writing the function make sure to type hint it correctly, so it can only get the types it is expected to.

4. Run the code against a user with the name of “Jane” and against another user with the name of “Bob”.

Understand Trait

Step1: Write the trait below in a file

```
trait Price {
public function changePriceByDollars($price, $change) {
return $price + $change; }
}
```

Step2: Use the traits

```
class Bmw {
        use Price;
}
class Mercedes {
        use Price;
}
```

Run the codes below

```
$bmw1 = new Bmw();
echo $bmw1 -> changePriceByDollars(45000, +3000);
$mercedes1 = new Mercedes();
echo $mercedes1 -> changePriceByDollars(42000, -2100);
```

Result: ??

Code Exercise

1. Create another trait

```
        trait SpecialSell {
                public function annonunceSpecialSell () {
                return __CLASS__ . " on special sell";
        }
}
```

2. Use in another class like below

```
class Mercedes {
        use Price;
        use SpecialSell;
}
$mercedes1 = new Mercedes();
// Subtract 2100$

echo $mercedes1 -> changePriceByDollars(42000, -2100);echo $mercedes1 ->
annonunceSpecialSell();
```

1 Which of the following keywords is used to define a trait?

A : defines

B : traits

C : trait

2 Which of the following types can be found within a trait?

A : abstract methods

B : concrete methods and properties

C : constants

D : all of the above

3 Is it appropriate to use traits in a scenario in which all of the child classes of an interface need the same methods?

Coding exercise

In the following exercise, we are going to write a simple program that uses an interface. In it, we are going to have three classes that inherit from the User interface: Author, Commentator, and Viewer. Only the first two classes have the need to use a Writing trait.

1 Write an interface with the name of User.

2 Write three classes to implement the User interface: Author, Commentator and Viewer.

3. Add a trait with the name of Writing that contains an abstract method with the name of writeContent().

4. Use the trait Writing in the Author class, and implement its abstract method by making it return the string “Author, please start typing an article...”.

5. Use the trait Writing in the Commentator class, and implement its abstract method by making it return the string “Commentator, please start typing your comment...”.

6. Run the code you have just written and see the result.