OOPS Exercise – Part 1

**Note**: your file name and path must follow below pattern

**Folder name:** php\_oops

**File name:** exe1.php, exe2.php etc...

**So your exercise location will look like** “domain/<your\_training\_folder\_path>/php\_oops/day1\_exe1.php”

## **Exercise 1**

Create one class “Employee”

Declare below properties –

* First name
* Last name
* Email
* Phone Number
* Birthdate
* Gender

Create individual set methods to set value of each property.

Create one getemployeedetails() method which should return all details of employee.

**Description**: Create a form which get above properties input from user and when user submit it, display employee information.

## **Exercise 2**

Build a class and store the data of the mobile phones with screen size, ram, company and processor along with Constructor and function to estimate cost of phone based on conditions

samsung:7000(pr)

moto:5999

lenovo:4000

nokia:6100

estimation cost = $pr\*($screenSize/4.0)+$pr\*($ram/1.0)+$pr\*($proccesor/1.0)+0.05\*$pr;

**Input**

enter screen size:5

enter ram:3

enter company name: moto

enter processor:3

**Output**

price:43792

Create class Mobile and declare below properties:

* Screen size
* Ram
* Company
* Processor

Use constructor to set value of properties.

Method:

* setprice – to calculate price from given details.
* getprice – to get price.

## **Exercise 3 (Inheritance Usage)**

Make form which contains bank account details inputs.

* Account number
* Account balance
* Account Type: Saving account and Current Account. (Dropdown to select)
* Give textbox add interest for both account type.

On submit Account number, Account balance entered, Account type and updated account balance should display.

**Create a Bank Account class.**

Create methods for set amount, set account number, get amount and get account number, account summary in Bank class.

Create saving account and current account subclasses which extends bank account class. Create method to “add interest” in saving account class and current account both class individually.

Bank account class will have Account number, Account balance details which will be inherited by subclasses.

## **Exercise 4 (Abstract Class Usage)**

Define the following classes and follow the instructions carefully to pass the tests.

**Person**: This class should be declared abstract

**Properties**: $name, $age, and $gender (all public)

**Constructor**: The class constructor should receive three required arguments in the following order: $name, $age, $gender and should set their respective properties.

**Methods:**

* **introduce**: Every person knows how to introduce himself/herself in some way (well, maybe except for babies but we won't be dealing with them in this Kata), but different types of people can introduce themselves in very different ways so there is no universal implementation of the introduce method. Therefore, declare this method abstract. This method should receive no arguments.
* **greet**: This method should receive exactly one required argument $name and return a string of the form "Hello NAME\_HERE".

**Child**: Since a Child "is a" Person, the Child class should inherit from the Person class.

**Properties**: $name, $age, and $gender - public properties of “Person” class.

Child class should also have an additional public property $aspirations which is expected to be an array of strings

**Constructor**: The class constructor should receive four required arguments in the following order: $name, $age, $gender, $aspirations and set their respective properties correctly.

**Methods:**

* **introduce**: This public method should receive no arguments and return a string of the form "Hi, I'm NAME\_HERE and I am AGE\_HERE years old".
* **greet**: This public method should accept exactly one required argument $name and return a string of the form "Hi NAME\_HERE, let's play!".
* **say\_dreams:** This public method should accept no arguments and return a string of the form "I would like to be a(n) ASPIRATIONS\_HERE when I grow up."

**Software Developer:** Since a Software Developer "is a" Person, the Software Developer class should inherit from the Person class.

**Properties:** $name, $age, and $gender – all public properties of “Person” class.

Software Developer class should also have an additional public property $position which is expected to be a string

**Constructor:** The class constructor should receive four required arguments in the following order: $name, $age, $gender, $position and set their respective properties correctly.

**Methods:**

* **introduce**: This public method should accept no arguments and return a string of the form "Hello, my name is NAME\_HERE, I am AGE\_HERE years old and I am a(n) OCCUPATION\_HERE"
* **greet**: This public method should accept an argument $name and return a string of the form "Hello , I'm OWN\_NAME\_HERE, nice to meet you".
* **describe job**: This public method should accept no arguments and return a string of the form "I am currently working as a(n) OCCUPATION\_HERE"

User interface implementation to use above structure is described as below

Create one form with below user inputs

* **I am a**: Dropdown of “Child” and “Software Developer”
* **My Name**: Textbox to enter name
* **My Age**: Textbox to enter age
* **Gender**: Dropdown of “Male” and “Female”
* **My Dreams (Visible only If you have selected “Child”)**: checkbox with list of “Doctor”,“Engineer”,“Lawyer” and “Police Officer”
* **My Position (Visible only If you have selected “Software developer”)**: “Backend Developer”,“Front End Developer”, “Full Stack Developer”

**On submission of form all details must be shown using appropriate class methods.**