Coupling/Messages

Coupling refers to dependencies of classes. Some classes can have dependencies (they can require other classes in order to be instantiated):

 class Person

 {

     protected $name;

     public function \_\_construct($name)

     {

         $this->name = $name;

     }

 }

 class Business

 {

    protected $staff;

    //the business class needs an instance of the Staff class when it's initialized (so it can add the person it employs to the staff collection)

    public function \_\_construct(Staff $staff)

    {

        $this->staff = $staff;

    }

    //the hire() method takes in as parameter only an instance of the Person class (this is called type hinting)

    public function hire(Person $person){

        //add person to the staff collection

        $this->staff->add($person);

    }

 }

 class Staff

 {

    protected $members = [];

    public function add(Person $person)

    {

        //add the person to the members array

        $this->members[] = $person;

    }

 }

\*the Business class requires an instance of the Staff class when it is instantiated (see its constructor)

\*classes communicate with one another through sending ‘messages’ (one can require the use of another’s instance/methods just like the Staff instance requires a Person instance and the Buisness instance requires a Staff instace). Sending messages means that one class can call a method from the instance of another class which is being passed as parameter to it(e.g. Business instance calls a method of the Staff class in one of its own methods);

\*also the hire() method in the Staff class, requires a Person object:

//the hire() method takes in as parameter only an instance of the Person class (this is called type hinting)

    public function hire(Person $person){

        //add person to the staff collection

        $this->staff->add($person);

    }

In order to better the Staff class, we can make it so that when it is instantiated, it must have a $members argument (either passed in or defined as default):

 class Staff

 {

    protected $members = [];

    public function \_\_construct($members = [])

    {

        $this->members = $members;

    }

    public function add(Person $person)

    {

        //add the person to the members array

  array\_push($this->members, $person);

    }

 }

\*now the Staff class can have $members as passed in argument when instantiated:

 $staff = new Staff($jeffrey);

\*if no $members argument is passed in, the default empty array [] will be used;

In order not to have to use the $staff variable all the time, we can make a method of getStaffMembers() in the Company class (method which will call a members() method in the Staff class):

   class Business

    {

        protected $staff = [];

        public function \_\_construct(Staff $staff)

        {

            $this->staff = $staff;

        }

        public function hire(Person $person)

        {

            $this->staff->add($person);

        }

        public function getStaffMembers()

        {

            return $this->staff->members();//calls members() from Staff

        }

    }

    class Staff

    {

        protected $members = [];

        //set $members default to empty array (but $members can also be passed through as argument when instantiating a new Staff obj)

        public function \_\_construct($members =[])

        {

            $this->members[] = $members;

        }

        public function add (Person $person)

        {

            array\_push($this->members, $person);

        }

        public function members()

        {

            return $this->members; //called by getStaffMembers() in Business

        }

    }

\*messages means that classes communicate to one another by calling each other’s methods (like Business class calls the members() method of the Staff class in its own getStaffMembers() method;

\*similarly the hire() method in Staff calls the add() method in Staff;