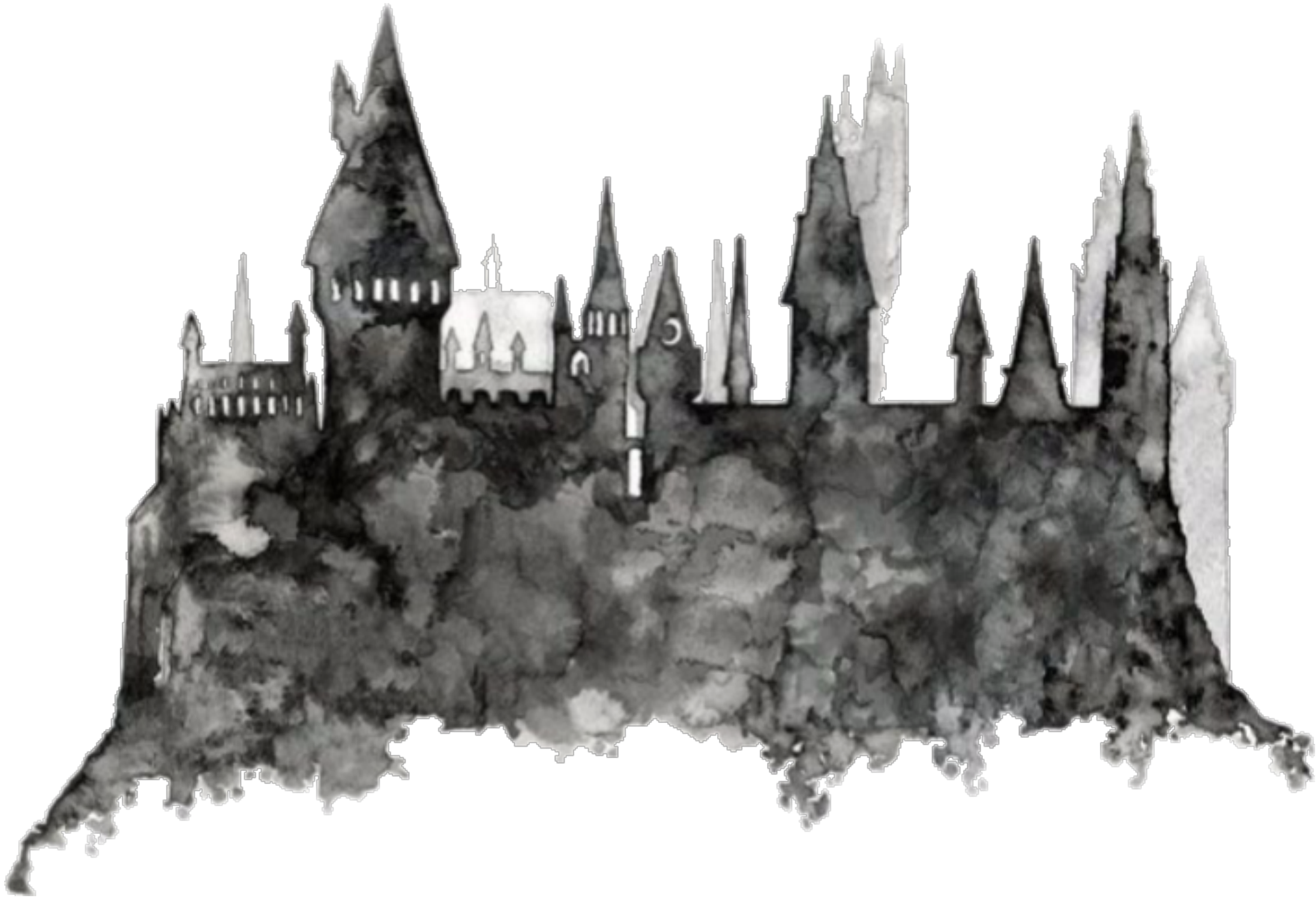
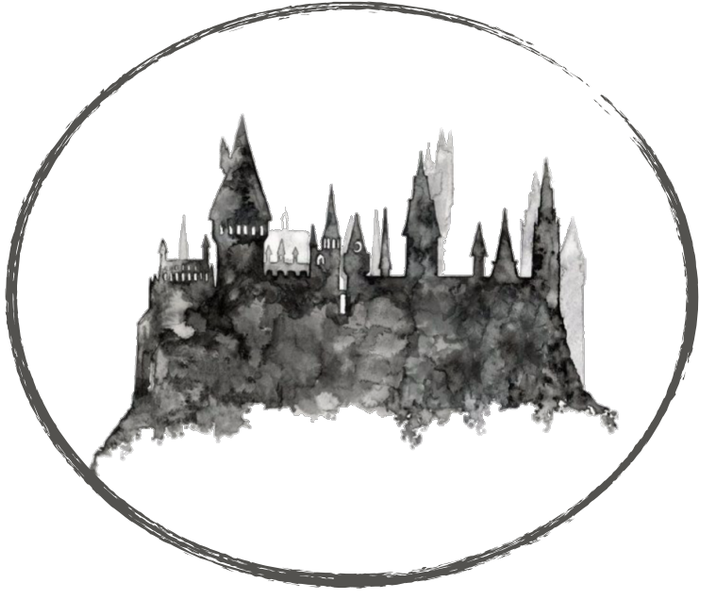
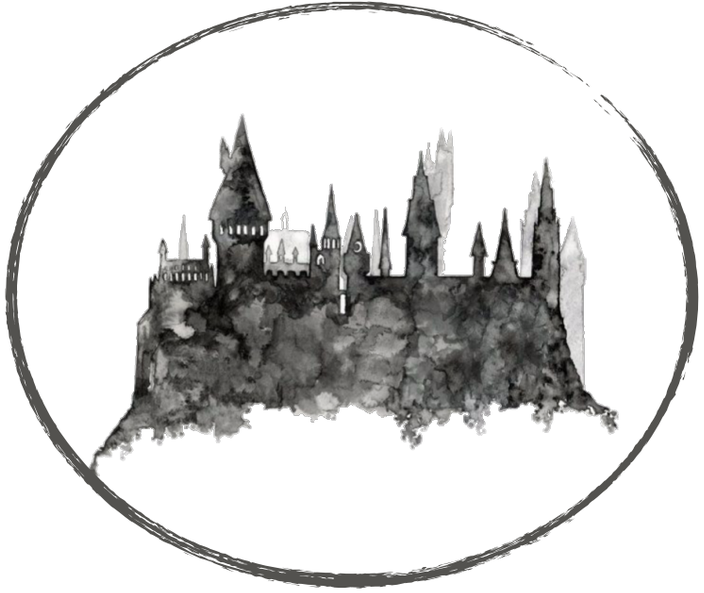
Once upon a time



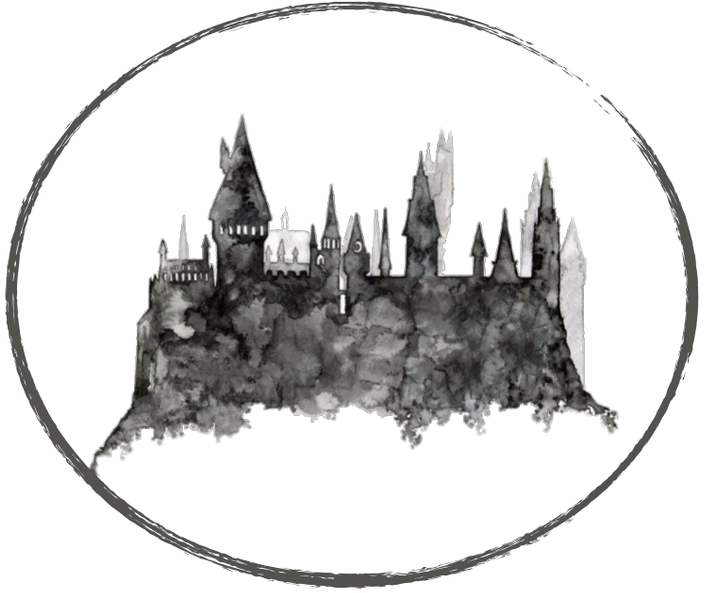
There was a wizarding school called Hogwarts



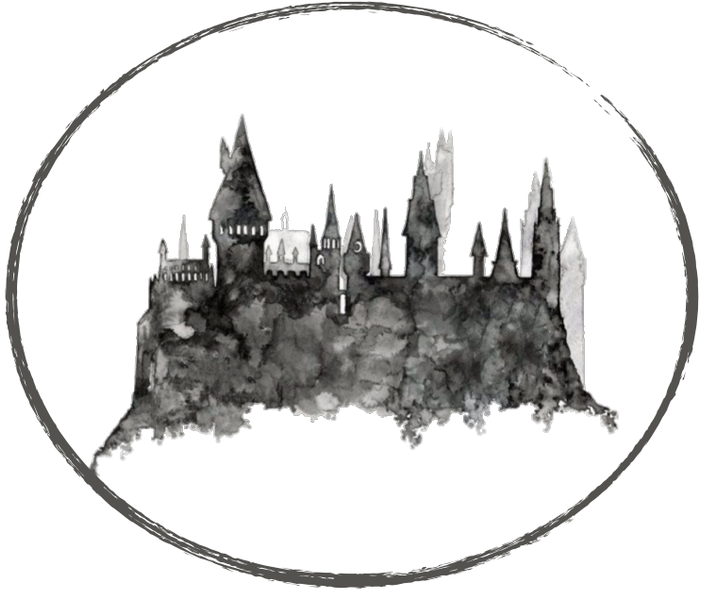
bearing the last name of its founder



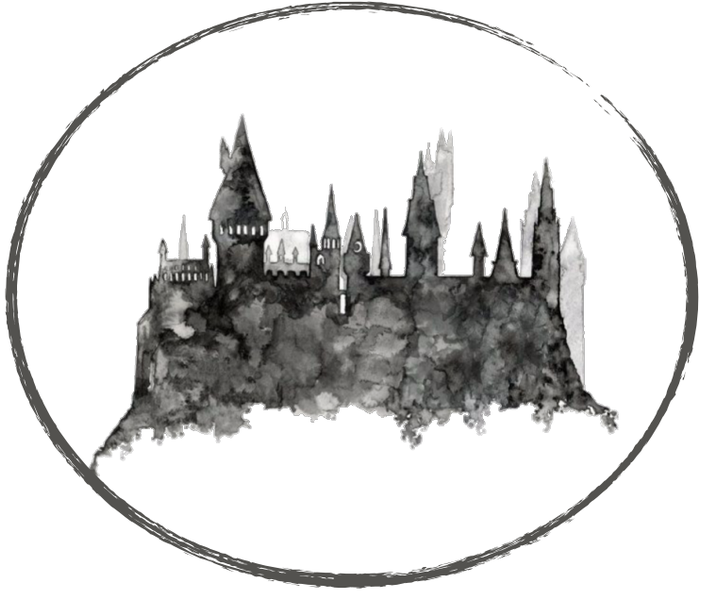
Godric Gryffindor,



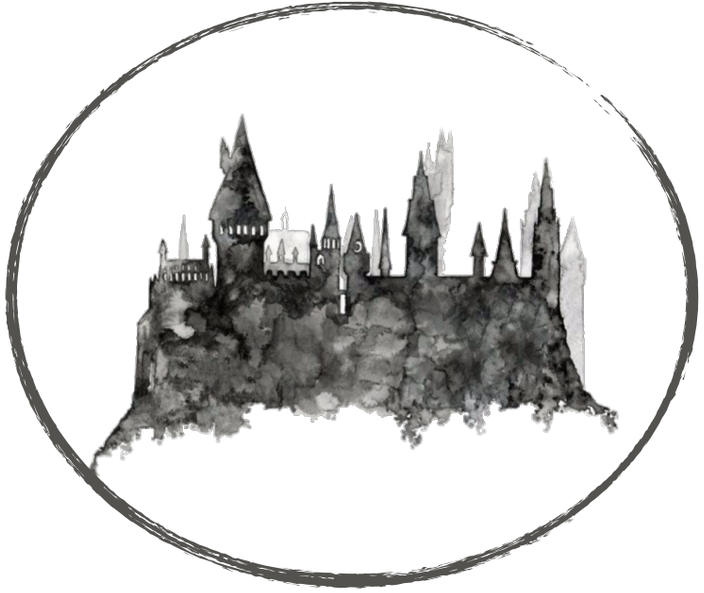
Salazar Slytherin,



Rowena Ravenclaw



and Helga Hufflepuff.



Inheritance

abstract class

Inheritance

abstract class

Daily Routines

	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 – 9:45	Transfiguration	Potions	DADA	Charms	Divination
9:45 – 10:30					
10:30 – 10:45	Break	Break	Break	Break	Break
10:45 – 11:30	Muggle Studies	Spare	Ancient Runes	Arithmancy	CoMC
11:30 – 12:15	Potions	Transfiguration	Herbology	DADA	Charms
12:15 – 1:00			HoM		
1:00 – 2:00	Lunch	Lunch	Lunch	Lunch	Lunch
2:00 – 2:45	Games	CoMC	Arithmancy	Spare	HoM
2:45 – 3:30					Herbology
3:30 – 4:15	Ancient Runes	Ancient Runes	Muggle Studies	Divination	Muggle Studies
4:15 – 6:15	Dinner	Dinner	Dinner	Dinner	Dinner
6:15 – 7:00	Spare	Spare	Spare	Spare	Spare
7:00 – 7:45					
7:45 – 8:30					
8:30 – 9:15					
9:15 – 10:00					Astronomy

In Hogwarts , the first classe begins at 9h

Daily Routines

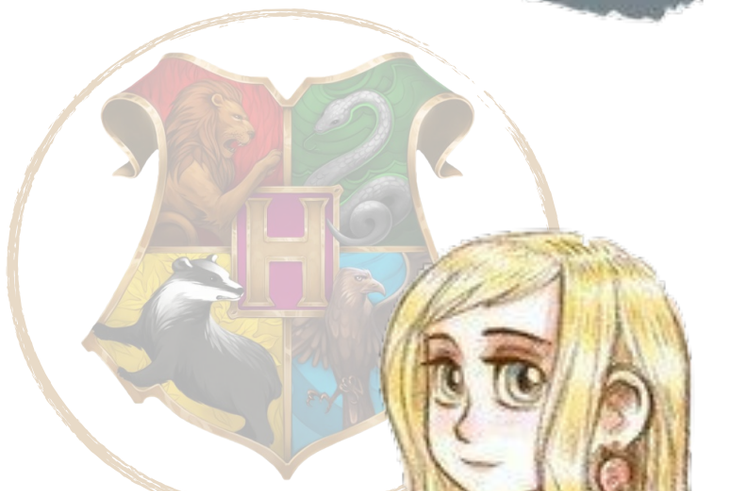
CONST

	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 – 9:45	Transfiguration	Potions	DADA	Charms	Divination
9:45 – 10:30					
10:30 – 10:45	Break	Break	Break	Break	Break
10:45 – 11:30	Muggle Studies	Spare	Ancient Runes	Arithmancy	CoMC
11:30 – 12:15	Potions	Transfiguration	Herbology	DADA	Charms
12:15 – 1:00			HoM		
1:00 – 2:00	Lunch	Lunch	Lunch	Lunch	Lunch
2:00 – 2:45	Games	CoMC	Arithmancy	Spare	HoM
2:45 – 3:30					Herbology
3:30 – 4:15	Ancient Runes	Ancient Runes	Muggle Studies	Divination	Muggle Studies
4:15 – 6:15	Dinner	Dinner	Dinner	Dinner	Dinner
6:15 – 7:00	Spare	Spare	Spare	Spare	Spare
7:00 – 7:45					
7:45 – 8:30					
8:30 – 9:15					
9:15 – 10:00					Astronomy

all students must be in their dormitories by 21h

Inheritance

abstract class



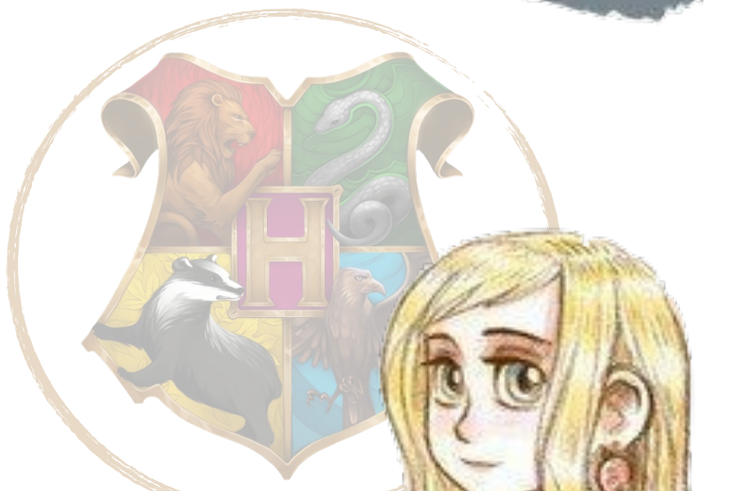
	Monday	Tuesday	Wednesday	Thursday	
9:00 – 9:45	Transfiguration	Potions	DADA	Charms	
9:45 – 10:30					
10:30 – 10:45	Break	Break	Break	Break	
10:45 – 11:30	Muggle Studies	Spare	Ancient Runes	Arithmancy	
11:30 – 12:15	Potions	Transfiguration	Herbology	DADA	
12:15 – 1:00			HoM		
1:00 – 2:00	Lunch	Lunch	Lunch	Lunch	
2:00 – 2:45	Games	CoMC	Arithmancy	Spare	
2:45 – 3:30					
3:30 – 4:15	Ancient Runes	Ancient Runes	Muggle Studies	Divination	
4:15 – 6:15	Dinner	Dinner	Dinner	Dinner	
6:15 – 7:00	Spare	Spare	Spare	Spare	
7:00 – 7:45					
7:45 – 8:30					
8:30 – 9:15					
9:15 – 10:00					

The Sorting ceremony was held

Inheritance

abstract class

CONST



	Monday	Tuesday	Wednesday	Thursday	
9:00 – 9:45	Transfiguration	Potions	DADA	Charms	
9:45 – 10:30					
10:30 – 10:45	Break	Break	Break	Break	
10:45 – 11:30	Muggle Studies	Spare	Ancient Runes	Arithmancy	
11:30 – 12:15	Potions	Transfiguration	Herbology	DADA	
12:15 – 1:00			HoM		
1:00 – 2:00	Lunch	Lunch	Lunch	Lunch	
2:00 – 2:45	Games	CoMC	Arithmancy	Spare	
2:45 – 3:30					
3:30 – 4:15	Ancient Runes	Ancient Runes	Muggle Studies	Divination	
4:15 – 6:15	Dinner	Dinner	Dinner	Dinner	
6:15 – 7:00					
7:00 – 7:45	Spare	Spare	Spare	Spare	
7:45 – 8:30					
8:30 – 9:15					
9:15 – 10:00					stronomy

After students had arrived at school

Inheritance

abstract class

CONST

	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 – 9:45	Transfiguration	Potions	DADA	Charms	Divination
9:45 – 10:30					
10:30 – 10:45	Break	Break	Break	Break	Break
10:45 – 11:30	Muggle Studies	Spare	Ancient Runes	Arithmancy	CoMC
11:30 – 12:15	Potions	Transfiguration	Herbology	DADA	Charms
12:15 – 1:00			HoM		
1:00 – 2:00	Lunch	Lunch	Lunch	Lunch	Lunch
2:00 – 2:45	Games	CoMC	Arithmancy	Spare	HoM
2:45 – 3:30					Herbology
3:30 – 4:15	Ancient Runes	Ancient Runes	Muggle Studies	Divination	Muggle Studies
4:15 – 6:15	Dinner	Dinner	Dinner	Dinner	Dinner
6:15 – 7:00	Spare	Spare	Spare	Spare	Spare
7:00 – 7:45					
7:45 – 8:30					
8:30 – 9:15					Astronomy
9:15 – 10:00					

After students had arrived at school

Inheritance

abstract class

CONST

	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 – 9:45	Transfiguration	Potions	DADA	Charms	Divination
9:45 – 10:30					
10:30 – 10:45	Break	Break	Break	Break	Break
10:45 – 11:30	Muggle Studies	Spare	Ancient Runes	Arithmancy	CoMC
11:30 – 12:15	Potions	Transfiguration	Herbology	DADA	Charms
12:15 – 1:00			HoM		
1:00 – 2:00	Lunch	Lunch	Lunch	Lunch	Lunch
2:00 – 2:45	Games	CoMC	Arithmancy	Spare	HoM
2:45 – 3:30					Herbology
3:30 – 4:15	Ancient Runes	Ancient Runes	Muggle Studies	Divination	Muggle Studies
4:15 – 6:15	Dinner	Dinner	Dinner	Dinner	Dinner
6:15 – 7:00	Spare	Spare	Spare	Spare	Spare
7:00 – 7:45					
7:45 – 8:30					
8:30 – 9:15					Astronomy
9:15 – 10:00					

After students had arrived at school

Inheritance

abstract class

CONST

Late Static Bindings

```
function whichHouse()
```

```
function whichHouse()
```

Before sorting,
a student doesn't
belong
to any house

After sorting,
a student
belongs
to a house

	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 – 9:45	Transfiguration	Potions	DADA	Charms	Divination
9:45 – 10:30					
10:30 – 10:45	Break	Break	Break	Break	Break
10:45 – 11:30	Muggle Studies	Spare	Ancient Runes	Arithmancy	CoMC
11:30 – 12:15	Potions	Transfiguration	Herbology	DADA	Charms
12:15 – 1:00			HoM		
1:00 – 2:00	Lunch	Lunch	Lunch	Lunch	Lunch
2:00 – 2:45	Games	CoMC	Arithmancy	Spare	HoM
2:45 – 3:30					Herbology
3:30 – 4:15	Ancient Runes	Ancient Runes	Muggle Studies	Divination	Muggle Studies
4:15 – 6:15	Dinner	Dinner	Dinner	Dinner	Dinner
6:15 – 7:00	Spare	Spare	Spare	Spare	Spare
7:00 – 7:45					
7:45 – 8:30					
8:30 – 9:15					
9:15 – 10:00					Astronomy

Summary

class Hogwarts

Attributes

```
protected $time,$whichHouse;
private $_routine;
const FIRSTCLASS = 9;
const DINNER = 16;
const CURFEW = 21;
```

Methods

```
public function whichHouse()
public function hatSorting()
public function schoolBell()
public function goToForbiddenForest()
```

abstract class House

Attributes

Methods

Inheritance

abstract class

CONST

Late Static Bindings

Summary

class Hogwarts

Attributes

```
protected $time,$whichHouse;
private $_routine;
const FIRSTCLASS = 9;
const DINNER = 16;
const CURFEW = 21;
```

Methods

```
public function whichHouse()
public function hatSorting()
public function schoolBell()
public function goToForbiddenForest()
```

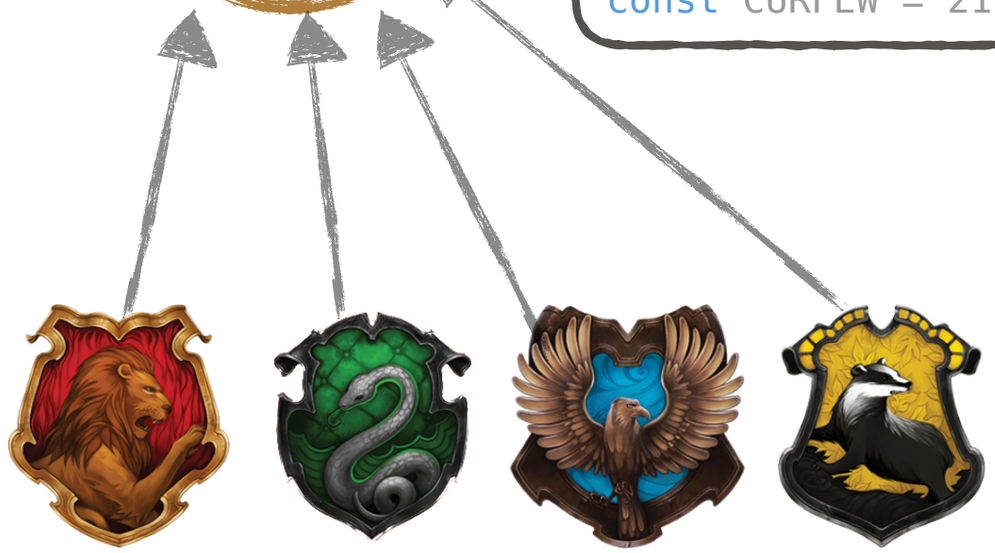
abstract class House

Attributes

```
protected $time,$whichHouse;

const FIRSTCLASS = 9;
const DINNER = 16;
const CURFEW = 21;
```

Methods



Inheritance

abstract class

CONST

Late Static Bindings

Summary

class Hogwarts

Attributes

```
protected $time,$whichHouse;
private $_routine;
const FIRSTCLASS = 9;
const DINNER = 16;
const CURFEW = 21;
```

Methods

```
public function whichHouse()
public function hatSorting()
public function schoolBell()
public function goToForbiddenForest()
```

abstract class House

Attributes

```
protected $time,$whichHouse;
private static $_houseCounter
const FIRSTCLASS = 9;
const DINNER = 16;
const CURFEW = 21;
```

Methods

```
public static function getCounter()
public function hatSorting()
```

Inheritance

abstract class

CONST

Late Static Bindings

protected

private

static

Let's Code

Classes defined as abstract cannot be instantiated.

An abstract Class is like a partially built class. It is much like a document with blanks to fill in.

When you extend a class, the subclass inherits all of the public and protected methods from the parent class.

Create a new instance of the class Gryffindor, $HarryPotter is an object.

```php
<?php

class Hogwarts
{

}


abstract class House extends Hogwarts
{

}


class Gryffindor extends House
{
}


$HarryPotter = new Gryffindor();
```

Inheritance

abstract class

instance

object

protected

private

const

getter

setter

Late Static Bindings

Only the school can plan the daily routine
➜ private

The houses and the school are on the same time zone
➜ protected

A Hogwarts student belongs necessarily to a house
➜ protected

As the name suggests, that value cannot change during the execution of the script
➜ const

```php
<?php
class Hogwarts
{
//--------------------- the attributes -------------------------
    private $_routine;


    protected $time,$whichHouse;


    const FIRSTCLASS = 9;
    const DINNER = 16;
    const CURFEW = 21;
}


abstract class House extends Hogwarts
{
}

class Gryffindor extends House
{
}

$HarryPotter = new Gryffindor();
```

Here, the attribute $time is protected, members declared protected can be accessed only within the class itself and by inheriting and parent classes.

A student is an instance of the class Hogwarts / an House, he have no access to the $time, but he still want to know what time is it. We need something public but not editable !


```php
<?php
class Hogwarts
{
// ----------------------- the attributes ---------------------
    protected $time,$whichHouse;
    private $_routine;
    const FIRSTCLASS = 9;
    const DINNER = 16;
    const CURFEW = 21;

//------------------------ List of getters-------------------
```

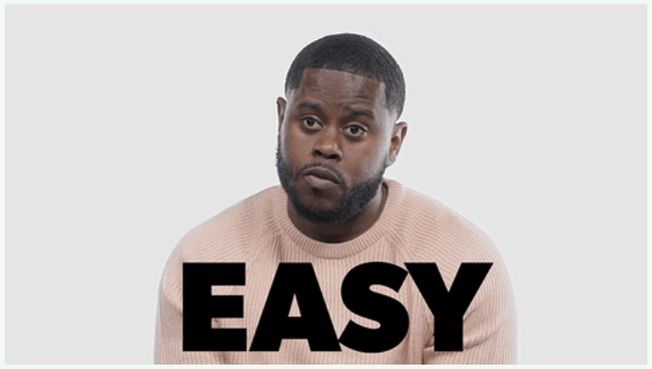
```php
}


abstract class House extends Hogwarts
{
}

class Gryffindor extends House
{
}

$HarryPotter = new Gryffindor();
```

“Getters” and “Setters” are object methods that allow you to control access to a certain class variables / properties.




```php
<?php
class Hogwarts
{
// ---------------------- the attributes --------------------
    protected $time,$whichHouse;
    private $_routine;
    const FIRSTCLASS = 9;
    const DINNER = 16;
    const CURFEW = 21;

//-------------------------- List of getters-------------------------

    public function time()
    {
            return $this->time;
    }


}


abstract class House extends Hogwarts
{
}

class Gryffindor extends House
{
}

$HarryPotter = new Gryffindor();
```

Solution : Getters

Inheritance

abstract class

instance

object

protected

private

const

getter

setter

Late Static Bindings

Now we need to set the security shield : Setters

“Getters” and “Setters” are object methods that allow you to control access to a certain class variables / properties.

```php
<?php
class Hogwarts
{
// ---------------------- the attributes --------------------
    protected $time,$whichHouse;
    private $_routine;
    const FIRSTCLASS = 9;
    const DINNER = 16;
    const CURFEW = 21;

//------------------------ List of getters------------------
    public function time()
    {
        return $this->time;
    }

   public function setTime($time)
     {
          if (!is_int($time))
          {
            trigger_error('time must be an integer', E_USER_WARNING);
            return;
          }
          if (is_int($time) && $time >= 0 && $time <= 23)
          {
          $this->time = $time;
          }
     }


}
abstract class House extends Hogwarts
{
}
```

The function whichHouse() returns the name of the house a student belongs.

We don’t want to rewrite this function for every child class of house.

Why not call hatSorting() directly?

Because we don’t have the sorting ceremony everyday!

➜ Late Static Bindings

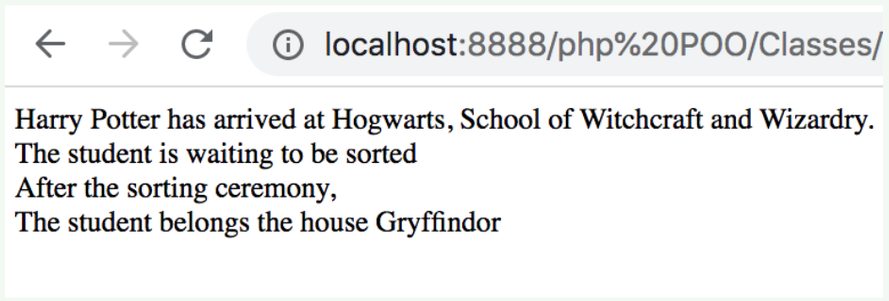
```php
<?php
class Hogwarts
{
//---------------------- the attributes ------------------------
    private $_routine;
    protected $time,$whichHouse;
    const FIRSTCLASS = 9;
    const DINNER = 16;
    const CURFEW = 21;
//--------------------- List of methodes ----------------------'
   public function whichHouse()
   {
          static::hatSorting();
   }

   public function hatSorting()
   {
          echo 'The student is waiting
to be sorted';
   }


}
abstract class House extends Hogwarts
{
   public function hatSorting()
   {
          echo ' The student belongs the
house '.get_called_class ();
   }
}


class Gryffindor extends House
```

Late Static Bindings



What happened with the code ?

```php
$HarryPotter = new Gryffindor();
$HarryPotter->whichHouse();
```

1, Call whichHouse() in Gryffindor
2, The method has not been rewritten.
3, Call whichHouse() in the parent class House.
4, The method has not been rewritten.
5, Call whichHouse() in the grandparent class Hogwarts, where we meet the function static::hatSorting() which calls the element of the class (Gryffindor) that is called during execution.

```php
<?php
class Hogwarts
{
// ---------------------- the attributes ---------------------
    protected $time,$whichHouse;
    private $_routine;
    const FIRSTCLASS = 9;
    const DINNER = 16;
    const CURFEW = 21;

    public function whichHouse()
    {
        static::hatSorting();
    }

    public function hatSorting()
    {
        echo 'The student is waiting to be sorted';
    }
}

abstract class House extends Hogwarts
{
    public function hatSorting()
    {
        echo ' The student belongs the house '.get_called_class ();
    }
}

class Gryffindor extends House
{
}
echo 'Harry Potter has arrived at Hogwarts,
School of Witchcraft and Wizardry. <br />' ;
$HarryPotter = new Hogwarts();
$HarryPotter->whichHouse();
echo "<br />";
echo 'After the sorting ceremony, <br />' ;
$HarryPotter = new Gryffindor();
$HarryPotter->whichHouse();
```

Inheritance
abstract class
instance
object
protected
private
const
getter
setter

Late Static Bindings

to be continued