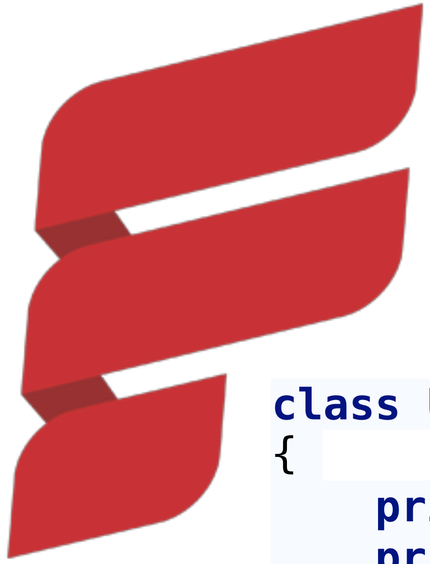
Exceptions

sd@freematiq.com



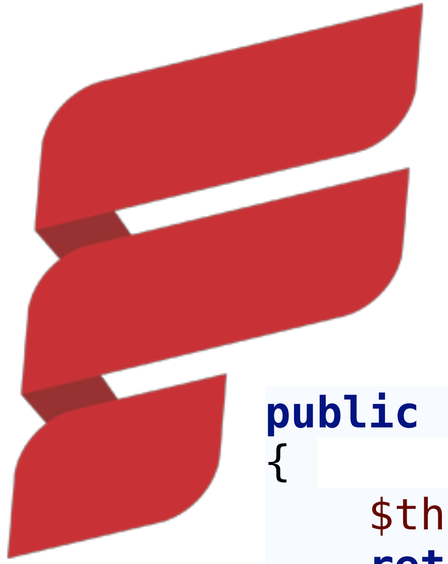
__toString

```php
class User
{
    private $name;
    private $password;

    public function __toString()
    {
        return "User. Name: {$this->name}" . PHP_EOL;
    }
}

$user = new \Freematiq\User();
echo $user;
```

Не используйте __toString() в бизнес логике!
Можно использовать для удобства



Цепочка вызовов

```php
public function setName($name)
{
    $this->name = $name;
    return $this;
}


public function setPassword($password)
{
    $this->password = $password;
    return $this;
}

$user = new \Freematiq\User();
$user->setName('Ivan')->setPassword('USSR');
echo $user;
```

Исключения

Исключение - некоторое критическое сообщение об ошибке

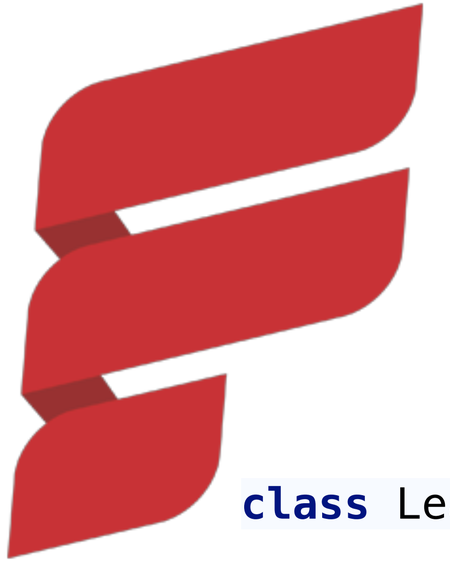
При генерации оно передаётся в ту часть программы, которая его обработает

Исключение - это объект

Содержит текст + системную информацию

Пример

```php
try {
    echo "В блоке исключения" . PHP_EOL;
    if (true) {
        throw new \Exception("Что-то пошло не так!");
    }
    echo "Всё хорошо!"; // уже не выполнится
} catch (\Exception $e) {
    echo "Oops!" . PHP_EOL;
    echo "Problem: {$e->getMessage()}" . PHP_EOL;
}
```



Наследование исключений

```php
class LearningException extends \Exception {}

try {
    echo "В блоке исключения" . PHP_EOL;
    if (true) {
        throw new LearningException("Что-то пошло не так!");
    }
    echo "Всё хорошо!"; // уже не выполнится
} catch (LearningException $e) {
    echo "Ой!" . PHP_EOL;
    echo "Наша учебная ошибка: {$e->getMessage()}" . PHP_EOL;
} catch (\Exception $e) {
    echo "Oops!" . PHP_EOL;
    echo "Problem: {$e->getMessage()}" . PHP_EOL;
}
```

Когда и как использовать

Все исключения состоят в иерархии
Exception - самый высокий уровень (в php 5)

Использовать при нештатных ситуациях, когда их обработка переносится на более высокий уровень

if оставить там, где обработка ситуации решается “на месте”

Exception src

```
public function __construct($message = "", $code = 0,
Throwable $previous = null) { }

class Exception implements Throwable {
    protected $message;
    protected $code;
    protected $file;
    protected $line;

    final public function getTrace() { }
    public function __toString() { }
```

e-mail

https://swiftmailer.symfony.com

composer require swiftmailer/swiftmailer

Swift_SmtpTransport - настройка отправки
Swift_Mailer - отправка почты
Swift_Message - сообщение

e-mail

```php
$transport = new
Swift_SmtpTransport('smtp.mailtrap.io', 25);

$transport
    ->setUsername('194152d5330321233')
    ->setPassword('3687a094fdbcbb');

$swift = new Swift_Mailer($transport);

$message = new Swift_Message('Всем привет!', 'Это письмо
отправлено из учебного класса');
$message->setFrom('sd@freematiq.com', 'Сергей');
$message->setTo('all@freematiq.com', 'Класс');

$result = $swift->send($message);

echo "Mail sent. Result = $result\n";
```

DateTime

С php 5.2

- DateTime - время
- DateTimeZone - временная зона
- DateInterval - интервал между двумя датами

DateTime

```php
$date = new DateTime();
echo $date->format('Y-m-d H:i:s'), PHP_EOL;
echo $date->getTimestamp(), PHP_EOL;

$date = new DateTime('now', new DateTimeZone('Europe/Moscow'));
echo $date->format('Y-m-d H:i:s'), PHP_EOL;

// public function setDate ($year, $month, $day) {}
$date->modify('+3 days');
echo $date->format('Y-m-d H:i:s'), PHP_EOL;

$date1 = new DateTime('2017-07-24 19:00');
$date2 = new DateTime('2017-01-01 0:00');

$result = $date2->diff($date1);
print_r($result);
```