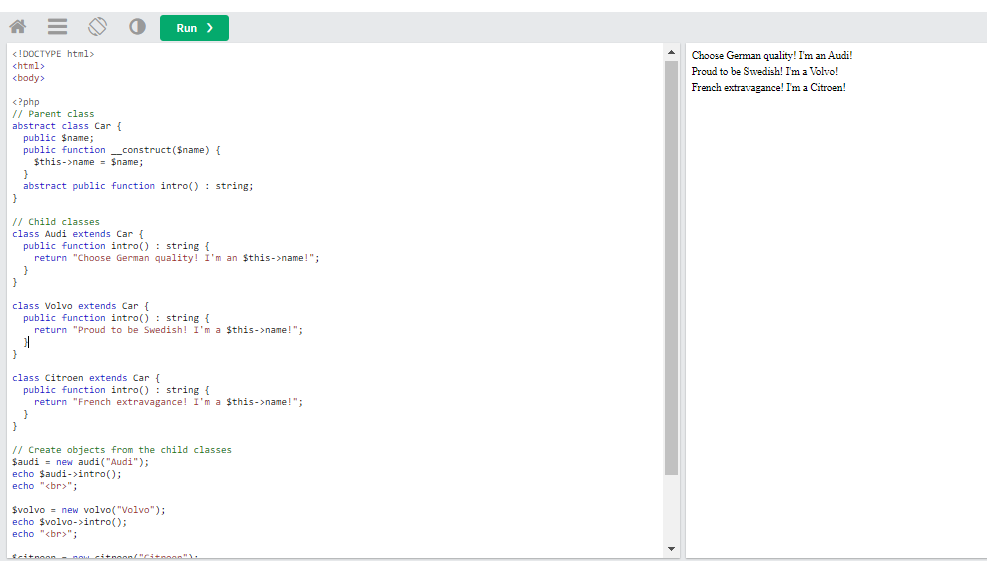
OOP

1. Abstract Classes:

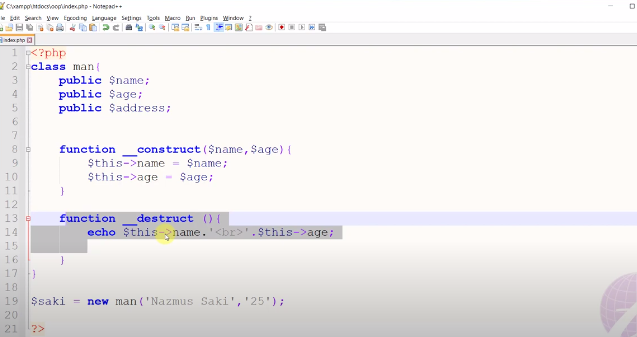
Parent ক্লাসের মধ্যে কোনো একটি মেথড ঘোষণা করেন কিন্তু সেই মেথরের কাজটি parent ক্লাস হবে না চাইল্ড ক্লাসে হবে। অর্থাৎ সে মেথরের ক্লাস executive হতে হবে যা একটি ক্লাসে



2.constructor: object create korar sathe AUTO excute hobe.call korar need hoy nai



1. Construct er kaj ses houyar pore auto destruct er vitor doke jay



3.

1. **Encapsulation ;** **মানে:** ডেটা (property) এবং ফাংশন (method) একত্রে একটি ক্লাসে প্যাক করে রাখা এবং বাইরের দুনিয়া থেকে ডেটা লুকানো।  
   **কেন:** ডেটা সুরক্ষিত রাখতে এবং শুধু method-এর মাধ্যমে access করতে দিতে।

**class BankAccount {**

**private $balance = 0; // balance সরাসরি access করা যাবে না**

**public function deposit($amount) {**

**if ($amount > 0) {**

**$this->balance += $amount;**

**}**

**}**

**public function getBalance() {**

**return $this->balance;**

**}**

**}**

**$account = new BankAccount();**

**$account->deposit(1000);**

**echo $account->getBalance(); // আউটপুট: 1000**

**ব্যাখ্যা:**  
$balance private, তাই বাইরে থেকে access করা যায় না। deposit() ও getBalance() method এর মাধ্যমে নিয়ন্ত্রণ করা হয়। এটি হলো encapsulation।

1. **Inheritance :** **মানে:** একটি ক্লাস অন্য ক্লাসের বৈশিষ্ট্য (property/method) গ্রহণ করে।  
   **কেন:** কোড পুনঃব্যবহারযোগ্য (reusable) করার জন্য।

class Animal {

public function sound() {

echo "Some generic sound";

}

}

class Dog extends Animal {

public function sound() {

echo "Bark!";

}

}

$dog = new Dog();

$dog->sound(); // আউটপুট: Bark!

🟢 **ব্যাখ্যা:**  
Dog ক্লাস Animal থেকে বৈশিষ্ট্য পেয়েছে। sound() method override করেছে।

### 3. **Polymorphism (পলিমরফিজম)**

**মানে:** একই method ভিন্নভাবে বিভিন্ন ক্লাসে কাজ করে।  
**কেন:** একাধিক ক্লাসে একই ইন্টারফেস মেইনটেইন করে behavior পরিবর্তন করা যায়।

class Animal {

public function makeSound() {

echo "Animal sound";

}

}

class Cat extends Animal {

public function makeSound() {

echo "Meow";

}

}

class Cow extends Animal {

public function makeSound() {

echo "Moo";

}

}

// ব্যবহার

function printSound(Animal $animal) {

$animal->makeSound();

}

printSound(new Cat()); // Meow

printSound(new Cow()); // Moo

🟢 **ব্যাখ্যা:**  
printSound() function একই টাইপের ইনপুট নিয়ে ভিন্নভাবে কাজ করছে। এটাই Polymorphism।

### 4. **Abstraction (অ্যাবস্ট্রাকশন)**

**মানে:** দরকারি জিনিস দেখানো এবং অপ্রয়োজনীয় জিনিস লুকানো।  
**কেন:** জটিলতা লুকিয়ে ইউজারের জন্য কাজ সহজ করা।

abstract class Shape {

abstract public function area(); // শুধু ঘোষণা, কোন বাস্তবায়ন নেই

}

class Rectangle extends Shape {

private $length;

private $width;

public function \_\_construct($l, $w) {

$this->length = $l;

$this->width = $w;

}

public function area() {

return $this->length \* $this->width;

}

}

$rect = new Rectangle(5, 4);

echo $rect->area(); // 20

**ব্যাখ্যা:**  
Shape ক্লাসে শুধু method এর নাম ঘোষণা আছে। কিভাবে কাজ করবে সেটা child class Rectangle define করেছে। এটি হলো abstraction।

## SOLID Principles

**SOLID** হলো ৫টি Object-Oriented Programming (OOP) নীতির সংক্ষিপ্ত রূপ:

| **Letter** | **নাম** | **বাংলা অর্থ** | **কাজ** |
| --- | --- | --- | --- |
| S | **Single Responsibility Principle** | একটি ক্লাসের একটি মাত্র দায়িত্ব থাকবে | এক ক্লাস = এক কাজ |
| O | **Open/Closed Principle** | কোড পরিবর্তন না করে নতুন ফিচার যোগ করা যাবে | কোড "open for extension, closed for modification" |
| L | **Liskov Substitution Principle** | Parent ক্লাস যেখানে কাজ করে, সেখানে Child ক্লাসও কাজ করবে | Subclass → Superclass behavior match |
| I | **Interface Segregation Principle** | ছোট ছোট নির্দিষ্ট ইন্টারফেস বানান | ক্লায়েন্ট যেন শুধু প্রয়োজনীয় method ইমপ্লিমেন্ট করে |
| D | **Dependency Inversion Principle** | High-level ক্লাস Low-level এর উপর নির্ভর করবে না | Interface বা abstract class ব্যবহার করে loose coupling করুন |

### // ❌ Bad: এক ক্লাসে অনেক দায়িত্ব

### class User {

### public function save() {}

### public function sendEmail() {}

### }

### // ✅ Good: প্রত্যেক ক্লাস আলাদা কাজ করছে

### class User {

### public function save() {}

### }

### class EmailService {

### public function sendEmail(User $user) {}

### }

## DRY Principle: **Don't Repeat Yourself**

📌 অর্থ: একই কোড বারবার লিখবেন না।

❌ খারাপ:

$total = $price \* $quantity;

$tax = $price \* $quantity \* 0.15;

; ✅ ভালো:

$subtotal = $price \* $quantity;

$tax = $subtotal \* 0.15

## KISS Principle: **Keep It Simple, Stupid**

📌 অর্থ: কোড যত সম্ভব **সহজ** রাখুন। জটিলতা এড়ান।

❌ খারাপ:

if ($status == true && $status !== false && $status === 1) { ... }

ভালো:

if ($status) { ... }

## YAGNI Principle: **You Aren’t Gonna Need It**

📌 অর্থ: ভবিষ্যতে লাগবে **বলে এখনি ফিচার লিখবেন না**।

❌ খারাপ: আগেই extra future-proofing

function uploadFile($file, $type = 'pdf', $resize = false, $compress = true) { ... }

✅ ভালো: এখন যেটা দরকার, সেটাই কোড করুন

function uploadPdf($file) { ... }

## সারসংক্ষেপ (OOP 4 Pillars):

| **OOP পিলার** | **ব্যাখ্যা** |
| --- | --- |
| ✅ Encapsulation | ডেটা লুকানো, access control (public/private) |
| ✅ Inheritance | একটি ক্লাস অন্য ক্লাস থেকে বৈশিষ্ট্য নেয় |
| ✅ Polymorphism | একই method, কিন্তু ভিন্ন আচরণ |
| ✅ Abstraction | শুধু দরকারি behavior define করা, details লুকানো |