**Member function and veriable**

ক্লাসের ভিতর ভেরিয়েবল কে বলা হয় প্রোপর্টিজ বা মেম্বার ভেরিয়েবল এবং ফাংশনকে বলা হয় মেথড বা মেম্বার ফাংশন বা (behaviour)বিহেবিয়ার।

**\_\_constructor():**

পিএইচপির বিশেষ ধরণের বিল্ট-ইন মেথড। অবজেক্ট তৈরির সময় প্রোপার্টির ভ্যালু এসাইনে সম্মতিদেয়। নতুন কোনো অবজেক্ট ক্রিয়েট করলে ক্লাসের মধ্যে যে মেথডটি সবার আগে এক্সেকিউট হয়, তাকে \_\_\_কন্সট্রাক্টর বলে।

**Function**

· Function হল Group of Code যা একটি নির্দিষ্ট কাজ করে । এক বা একাধিক ইনপুট প্যারামিটার গ্রহণ করতে পারে । একটি আউটপুট ভ্যালু রিটার্ন করতে পারে ।

**Function এবং Constructor এরপার্থক্য---**

|  |  |
| --- | --- |
| Function | Constructor |
| Function হল Group of Code যা একটি নির্দিষ্ট কাজ করে | Constructor এর কাজ হল class এর Instance তৈরি করা। |
| Function এর return type থাকে | Constructor এর কোন return type নেই এমনকি void ও ব্যবহার করা যায়না। |
| Function এ final, static, abstract এই ধরনের Modifier গুলো ব্যবহার হয় | Constructor এ এই ধরনের Modifier গুলো ব্যবহার করা যায় না। |
| Function এ this কিওয়ার্ড ব্যবহার হয় Working Class এবং Parent Class এর static and non static property,method কে Refer করার জন্য তবে Parent Class এর static and non static method কে parent,self কিওয়ার্ড দ্বারাও Refer করা যায়। | Constructor এ parent and self কিওয়ার্ড ব্যবহার হয় Parent Class এর Constructor কে Call করার জন্য। |

---------------------------------

**প্রোগ্রামিং ভাষায় কনটেক্সট কি ?**

---------------------------------

প্রোগ্রামিং ভাষায়, কনটেক্সট বলতে একটি স্পেসিফিক এনভায়রনমেন্ট বোঝায় যেখানে কোড কীভাবে কাজ করবে, কোথায় ব্যবহৃত হবে, এবং কী ধরনের ডেটা প্রিপায়ার করা হবে । কনটেক্সট গুরুত্বপূর্ণ কারণ এটি প্রোগ্রামারদের কোড বুঝতে এবং ডিবাগ করতে সাহায্য করে ।

কনটেক্সটের উদাহরণ:

একটি ভ্যারিয়েবলের মান

একটি ফাংশনের আর্গুমেন্ট

একটি ক্লাসের অবস্থা

একটি প্রোগ্রামের ইনপুট এবং আউটপুট

---------------------------------

**প্রোগ্রামিং ফিলোসফি কি ?**

---------------------------------

প্রোগ্রামিং ফিলোসফি হল প্রোগ্রামিং এর প্রিন্সিপল, প্রসিডিউর(পদ্ধতি) এবং টেকনিক(কৌশল) সম্পর্কে স্পষ্ট ধারণা দেয় যে, প্রোগ্রামাররা কীভাবে কোড লিখবে এবং কীভাবে সফ্টওয়্যার ডেভেলপমেন্ট করবে । প্রোগ্রামিং ফিলোসফি প্রোগ্রামারদের ভালো কোড লিখতে এবং উন্নত সফ্টওয়্যার ডেভেলপ করতে সাহায্য করে।

**প্রোগ্রামিং ফিলোসফি-র কিছু উদাহরণ: -----------------**

DRY (Don't Repeat Yourself): কোড রিউস করা ।

KISS (Keep It Simple Stupid): কোড যতটা সম্ভব সিম্পল রাখার ।

SOLID: মেইনটেইনেবল এবং ফ্লেক্সিবল সফ্টওয়্যার তৈরির জন্য SOLID প্রিন্সিপল ফলো করা ।

YAGNI (You Ain't/aren’t Gonna Need It): প্রয়োজন না হওয়া পর্যন্ত কোড না লেখার ।

**প্রোগ্রামিং আইসোলেট কি ?**

---------------------------------

প্রোগ্রামিং আইসোলেট হলো একটি প্রোগ্রামিং টেকনিক যেখানে একটি কোডকে নির্দিষ্ট স্কোপের মধ্যে আইসোলেট করে এবং অন্য কোডের সাথে ইন্টারঅ্যাকশন প্রিভেন্ট করে ।

**প্রোগ্রামিং আইসোলেট ব্যবহারের কারণ: -----------------**

Modularity: কোডকে মডিউল করে, মেইনটেইনএবল এবং টেস্টএবল করে তোলে।

Security: বাগ এবং সিকিউরিটি ভারনারেবিলিটির রিস্ক কমিয়ে দেয় ।

**প্রোগ্রামিং আইসোলেট তৈরি করার বিভিন্ন উপায়: -----------------**

Function: ফাংশনের মাধ্যমে কোডকে আইসোলেট করা যায় । একটি বেসিক ওয়ে ।

Class: ক্লাসের মাধ্যমে কোডকে আইসোলেট করা যায় । একটি পাওয়ারফুল ওয়ে ।

Module: মডিউল মাধ্যমে বড় ধরণের অ্যাপ্লিকেশনের কোডকে আইসোলেট করা যায় ।

---------------------------------

**প্রোগ্রামিং প্যারাডাইম(দৃষ্টান্ত) কি ?**

---------------------------------

প্রোগ্রামিং প্যারাডাইম হল প্রোগ্রাম লেখার উপায় । অর্থাৎ এটি নির্ধারণ করে যে প্রোগ্রাম কিভাবে লিখতে হবে এবং কীভাবে কাজ করে ।

**বিভিন্ন ধরণের প্রোগ্রামিং প্যারাডাইম রয়েছে: -----------------**

ইম্পারেটিভ প্রোগ্রামিং: এটি সবচেয়ে সাধারণ প্রোগ্রামিং প্যারাডাইম। এটি প্রোগ্রামার-কে নির্দেশ দেয় কিভাবে একটি নির্দিষ্ট কার্য সম্পাদন করতে হবে। C, C++, Java, Python

ডিক্লরেটিভ প্রোগ্রামিং: এটি প্রোগ্রামার-কে নির্দেশ দেয় কিভাবে একটি নির্দিষ্ট কার্য সম্পাদন করার জন্য ডেটা কে সংগঠিত করতে হবে। SQL, Prolog

প্রসিডিউরাল প্রোগ্রামিং: এটি প্রোগ্রামার-কে নির্দেশ দেয় কিভাবে একটি নির্দিষ্ট কার্য সম্পাদন করার জন্য ফাংশন গুলো কে সংযুক্ত করতে হবে। Haskell, Lisp

অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং: এটি প্রোগ্রামার-কে নির্দেশ দেয় কিভাবে একটি নির্দিষ্ট কার্য সম্পাদন করার জন্য অবজেক্ট গুলো কে ব্যবহার করতে হবে। Java, C++, Python

PHP প্রোগ্রামিং প্যারাডাইম: -----------------

PHP একটি মাল্টি-প্যারাডাইম প্রোগ্রামিং ল্যাংগুয়েজ, যার মানে বিভিন্ন প্রোগ্রামিং প্যারাডাইম ব্যবহার করে প্রোগ্রাম লেখ যায় । PHP তে নির্দিষ্ট কোন প্রোগ্রামিং প্যারাডাইম নেই । প্রোগ্রামার তাদের প্রয়োজন অনুযায়ী প্যারাডাইম গুলো নির্বাচন করতে পারে।

PHP এর সবচেয়ে সাধারণ প্রোগ্রামিং প্যারাডাইম হল: -----------------

প্রসিডিউরাল প্রোগ্রামিং: ফাংশন এবং প্রসিডিউর ব্যবহার করে প্রোগ্রাম লেখা হয় ।

অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং: অবজেক্ট এবং ক্লাস ব্যবহার করে প্রোগ্রাম লেখা হয় ।

---------------------------------

**প্রসিডিউরাল প্রোগ্রামিং কি ?**

---------------------------------

প্রসিডিউরাল প্রোগ্রামিং হলো এক ধরণের প্রোগ্রামিং প্যারাডাইম যেখানে ফাংশন এবং প্রসিডিউর ব্যবহার করে প্রোগ্রাম লেখা হয় ।

প্রসিডিউর: একটি নির্দিষ্ট কার্য সম্পাদন করে। একটি বা ততোধিক ইনপুট প্যারামিটার গ্রহণ করতে পারে । কোন আউটপুট ভ্যালু রিটার্ন করতে পারে না ।

সুবিধা: -----------------

প্রসিডিউরাল প্রোগ্রামিং শেখা এবং বোঝা সহজ ।

প্রসিডিউরাল প্রোগ্রামগুলি দ্রুত রান হয় ।

অসুবিধা: -----------------

প্রসিডিউরাল কোড প্রায়শই রিউস করা কঠিন ।

বড় প্রজেক্টের ক্ষেত্রে প্রসিডিউরাল কোড বোঝা কঠিন হতে পারে ।

প্রসিডিউরাল প্রোগ্রামগুলি বড় এবং জটিল প্রজেক্টের জন্য স্কেল করা কঠিন হতে পারে ।

**অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং কি ?**

---------------------------------

অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং হলো এক ধরণের প্রোগ্রামিং প্যারাডাইম যেখানে মেথড এবং প্রপার্টিকে অবজেক্ট এর মধ্যে অর্গানাইজ করে প্রোগ্রাম লেখা হয় ।

সুবিধা: -----------------

কোড চেঞ্জ করা সহজ ।

কোড রিইউজ করা সহজ ।

নতুন ফিচারস যোগ করা সহজ ।

অসুবিধা: -----------------

OOP এর কন্সেপ্টস গুলো শেখা জটিল হতে পারে ।

OOP তে প্রোগ্রামিং করলে সাধারণত বেশি মেমরি ব্যবহার হয় ।

**প্রসিডিউরাল প্রোগ্রামিং VS অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং ?**

|  |  |
| --- | --- |
| প্রসিডিউরাল প্রোগ্রামিং | অবজেক্ট ওরিয়েন্টেড |
| প্রসিডিউরাল প্রোগ্রামিং এ প্রসিডিউর গুলো গ্লোবাললি এক্সেসিবল, ফলে সহজে এক্সেস করা যায় | অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং এ একটা ক্লাস এর আন্ডারে মেথড এবং প্রপার্টি গুলো একত্রিত করা থাকে, ফলে সহজে এক্সেস করা যায় না |
| ডাটা হাইড নিয়ে কাজ করা যায় না | ডাটা হাইড নিয়ে কাজ করা যায় |
| There is no access specifier in procedural programming. | bject-oriented programming has access specifiers like private, public, protected, etc. |
| function greet($name) {  $message = "Hello, $name!";  echo $message;  }  $name = "Alice";  greet($name); // Output: Hello, Alice!  // Since there are no access specifiers, the variable $message within greet() is accessible here as well.  echo $message; // This might cause unintended side effects if not careful. | class Person {  private $name; // Private property, accessible only within the class  public function \_\_construct($name) {  $this->name = $name;  }  public function greet() {  $message = "Hello, " . $this->name . "!";  echo $message;  }  }  $alice = new Person("Alice");  $alice->greet(); // Output: Hello, Alice!  // Since $name is private, it cannot be accessed directly here.  // echo $alice->name; // This would cause an error. |
| In procedural programming, overloading is not possible. | Overloading is possible in object-oriented programming. |
| In procedural programming, the function is more important than the data. | Object-oriented programming is used for designing large and complex programs. |

**Classes ----ক্লাস কি ?**

১) ক্লাস হল একটি টেম্পলেট বা বুলুপ্রিন্ট যা অবজেক্ট তৈরির জন্য ব্যবহৃত হয় ।

২) ক্লাস অবজেক্টের স্ট্রাকচার (মেথড এবং প্রপার্টি) এবং কাস্টম টাইপ নির্ধারণ করে অথবা

Class is a programmer-defined data type.

৩) ক্লাস হল একটা স্ট্রাকচার (মেথড এবং প্রপার্টি) যেখানে স্পেসিফিক টাইপের একটা অবজেক্টে ক্রিয়েট করি ।

**অবজেক্ট কি ?**

অবজেক্ট হল ক্লাসের ইনস্ট্যান্স, যা স্পেসিফিক মেথড এবং প্রপার্টি ধারণ করে ।

**অবজেক্টকে প্রেসেন্ট করতে ৩ টি জিনিস লাগে: -----------------**

১) স্টেট/ attribute: অবজেক্টের ডাটা বা প্রপার্টি কে রিপ্রেজেন্ট করে ।

২) বিহেভিয়ার: অবজেক্টের অ্যাকশন বা মেথডকে রিপ্রেজেন্ট করে ।

৩) আইডেন্টিটি: অবজেক্টেকে ইউনিক করে তোলে । আইডেন্টিটি অন্যান্য অবজেক্ট থেকে একটি নির্দিষ্ট অবজেক্ট কে আলাদা করতে ব্যবহৃত হয়।

class Person {

public $name; // State

public $age; // State

public function sayHello() { // Behaviour

echo "Hello, my name is " . $this->name . " and I am " . $this->age . " years old.";

}

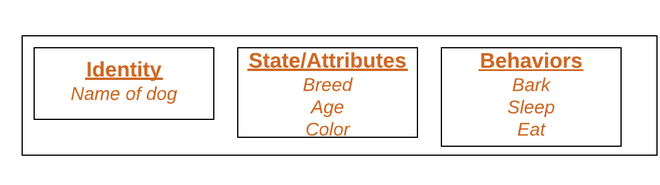
}

$person = new Person(); // Identity

$person->name = "John Doe";

$person->age = 30;

Object



**অবজেক্ট ক্রিয়েট করার দুটি উপায়: -----------------**

১) ক্লাস বেসড (অবজেক্ট বেসড): এটি পদ্ধতিটি জাভা, পিএইচপি এবং পাইথনের মতো ভাষায় ব্যবহৃত হয় ।

২) প্রোটোটাইপ বেসড: এই পদ্ধতিটি জাভাস্ক্রিপ্টের মতো ভাষায় ব্যবহৃত হয় ।

**ক্লাস বেসড এবং প্রোটোটাইপ বেসড মধ্যে নির্বাচন করা: --------**

১) ডেটা সিকিউরিটি এবং ইনহেরিটেন্স এর জন্য ক্লাস বেসড ।

২) ফ্লেক্সিবল এবং ডাইনামিক কোড এর জন্য প্রোটোটাইপ বেসড ।

ক্লাস প্রয়োজনীয়তা কেন ?

-----------------------------------

প্রোটোটাইপ বেসড প্রোগ্রামিং ক্ষেত্রে কোন নির্দিষ্ট স্ট্রাকচার ছাড়াই অবজেক্ট ক্রিয়েট করা যায় । অন্যদিকে, অবজেক্ট বেসড প্রোগ্রামিং ক্ষেত্রে অবজেক্ট তৈরি করার জন্য প্রথমে একটি স্ট্রাকচার নির্ধারণ করতে হয় যেখানে মেথড এবং প্রপার্টি গুলো থাকবে । এই স্ট্রাকচার কে ক্লাস বলা হয়, যার উপর বেস করে একটা অবজেক্টে ক্রিয়েট করা হয় ।

-----------------

**মেথড ওভারলোডিং:**

যখন একটি ক্লাসে একই নামের একাধিক মেথড থাকে তবে প্যারামিটার অলাদা হয় তাকে মেথড ওভারলোডিং বলে । PHP মেথড ওভারলোডিং সমর্থন করে, তবে অন্যান্য ল্যাঙ্গুয়েজের মত ওভারলোডিং সমর্থন করে না (যেমন, java বা C++)। PHP-তে মেথড ওভারলোডিং করার জন্য ম্যাজিক মেথড \_\_call বা \_\_callStatic মেথড ব্যবহার করা হয় ।

class Calculator {

// Magic method for method overloading

public function \_\_call($name, $arguments) {

if ($name === 'multiply') {

if (count($arguments) === 2) {

return $this->multiplyTwo($arguments[0], $arguments[1]);

} elseif (count($arguments) === 3) {

return $this->multiplyThree($arguments[0], $arguments[1], $arguments[2]);

}

}

throw new \BadMethodCallException("Method '$name' not found.");

}

function multiplyTwo(){

}

function multiplyThree(){

}

}

$calculator = new Calculator();

echo $calculator->add(5, 3) . PHP\_EOL; // Output: 8

echo $calculator->multiply(2, 3) . PHP\_EOL; // Output: 6

echo $calculator->multiply(2, 3, 4) . PHP\_EOL; // Output: 24

**মেথড ওভাররাইডিং:**

যখন প্যারেন্ট ক্লাস এবং চাইল্ড ক্লাসে এক্ই নামের মেথড থাকে যাদের প্যারামিটার(method signature) একই তখন তাকে মেথড ওভাররাইডিং বলে । এক্ষেত্ৰ চাইল্ড ক্লাসের মেথডেকে প্রায়রিটি বেশি । PHP মেথড ওভাররাইডিং সাপোর্ট করে

Method Overriding: -------------

class Animal {

public function makeSound() {

echo "Generic animal sound\n";

}

}

class Dog extends Animal {

public function makeSound() {

echo "Woof!\n";

}

}

class Cat extends Animal {

public function makeSound() {

echo "Meow!\n";

}

}

$dog = new Dog();

$dog->makeSound(); // Output: Woof!

$cat = new Cat();

$cat->makeSound(); // Output: Meow!

**অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং ফান্ডামেন্টাল কয়টা ?**

অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং এর ফান্ডামেন্টাল চারটি: -------------

১) অ্যাবস্ট্রাকশন

২) এনক্যাপসুলেশন

৩) ইনহেরিটেন্স

৪) পলিমরফিজম

অনেকের মতে অ্যাবস্ট্রাকশন একটা জেনেরিক কনসেপ্ট । এই কারণ অনেকে অ্যাবস্ট্রাকশনকে OOP ফান্ডামেন্টাল বলে না।

**অ্যাবস্ট্রাকশন কি ?**

অ্যাবস্ট্রাকশন হল ডিজাইন ফিলোসফি বা ডিজাইন আইডিয়ার। অর্থাৎ কনটেক্সট উপর বেস করে যখন কোনো ক্লাস ডিজাইন করি তখন ঐ ক্লাসে কি কি মেথড এবং প্রপার্টি থাকবে এবং কতটুকু রিপ্রেসেন্ট করবে বা কতটুকু এক্সপোস করবে এই ডিজাইন ফিলোসফি বা ডিজাইন আইডিয়ার হল অ্যাবস্ট্রাকশন । অ্যাবস্ট্রাকশন আননেসেসারি ইমপ্লিমেন্টেশন হাইড এবং শুধুমাত্র প্রয়োজনীয় ফাংশনালিটি শো/এক্সপোস করার ধারণা দেয় ।

উদাহরণ: ----------

১) একটি রিয়েল লাইফ এক্সাম্পল হল গাড়ি। যখন আমরা গাড়ি চালাই, আমরা ইঞ্জিন, গ্যাস সিস্টেম, ব্রেক সিস্টেম ইত্যাদির কীভাবে কাজ তা সম্পর্কে চিন্তা করি না। আমরা শুধুমাত্র গাড়ি চালানোর ধারণা রাখি, গাড়ির নকশা এই কমপ্লেক্স গুলিকে অ্যাবস্ট্রাক্ট করে, যাতে আমরা সহজেই গাড়ি চালাতে পারি।

২) একটি মানচিত্র হলো একটি রিয়েল ওয়ার্ল্ড অবস্ট্রাকশন। এটি রিয়েল ওয়ার্ল্ড এর সব বিবরণ দেয় না, শুধুমাত্র গুরুত্বপূর্ণ বিষয় যেমন রাস্তাঘাট, নদী, এবং ভবন গুলোর সম্পর্কে ধারণা দেয়।

3)টেলিফোন: ধরা যাক আপনি ফোন করতে চান। আপনার কি ইলেকট্রনিক সার্কিট, স্যুইচ এবং তারের জটিল নকশা সম্পর্কে জানতে হবে? অবশ্যই না! আপনি শুধু নম্বর ডায়াল করেন এবং ফোনটি আপনার কথা গ্রহীতার কাছে পৌঁছে দেয়। এটি অ্যাবস্ট্রাকশনের একটি দুর্দান্ত উদাহরণ। টেলিফোনের অভ্যন্তরীণ কার্যপ্রণালী লুকিয়ে রাখা হয়, যাতে আপনি সহজেই এটি ব্যবহার করতে পারেন।

**সুবিধা: ----------**

অ্যাবস্ট্রাকশন কোডের মূল কনসেপ্টগুলিকে ক্লিয়ার করে তোলে, ফলে কোডিং কমপ্লেক্সিটি কমে যায়।

আননেসেসারি ইমপ্লিমেন্টেশন ডিটেলস হাইড করে।

অসুবিধা: ----------

কমপ্লেক্স কনসেপ্টগুলির জন্য অ্যাবস্ট্রাকশন বাস্তবায়ন করা কঠিন হতে পারে।

অতিরিক্ত অ্যাবস্ট্রাকশন কোডকে অস্পষ্ট এবং বোঝার কঠিন করে তুলতে পারে, যার ফলে দক্ষতা হ্রাস পায়।

1. **অ্যাবস্ট্রাক্ট ক্লাস**:

এটি এমন এক ধরনের ক্লাস যা যেখানে এক বা একাধিক অ্যাবস্ট্রাক্ট এবং নন-আ্যবস্ট্রক্ট মেথড থাকে। অ্যাবস্ট্রাক্ট মেথডগুলির কোনো নির্দিষ্ট ইমপ্লিমেন্টেশন থাকে না, শুধুমাত্র ডিক্লারেশন থাকে। যে ক্লাস এই অ্যাবস্ট্রাক্ট ক্লাসটি ইমপ্লিমেন্ট করে, সে এই মেথডগুলির নির্দিষ্ট ইমপ্লিমেন্টেশন প্রদান করতে বাধ্য। Abstract Class এর অবজেক্ট create করা যায় না

**Abstract Method কি?**

Abstract method হল এমন এক ধরনের মেথড যার শুধু ঘোষণা (declaration) থাকে,বাস্তবায়ন(implementation) থাকেনা।

**এনক্যাপসুলেশন কি ?**

এনক্যাপসুলেশন বলতে কোনো কিছু আবদ্ধ অবস্থায় রাখাকে বোঝায় । অর্থাৎ একটা ক্লাসের মধ্যে মেথড এবং প্রপার্টি গুলোকে আবদ্ধ অবস্থায় রাখার প্রক্রিয়াকে এনক্যাপসুলেশন বলে । এনক্যাপসুলেশনের মাধ্যমে ক্লাসের বাইরে থেকে ক্লাসের মেথড এবং প্রপার্টি গুলোকে সরাসরি অ্যাক্সেস প্রিভেন্ট করা যায় । এক্সেস মডিফায়ার ব্যবহার করে এনক্যাপসুলেট করা হয় ।

**সুবিধা: ----------**

১) ডেটা সিকিউরিটি প্রোভাইড করে ।

২) ক্লাসের মধ্যে মেথড এবং প্রপার্টি গুলো (এক্সেস মডিফায়ার ব্যবহার করে ) কন্ট্রোল করা যায় ।

**অসুবিধা: ----------**

১) অতিরিক্ত এনক্যাপসুলেশন ফলে কোড বোঝা কঠিন হতে পারে ।

২) এনক্যাপসুলেশন ফলে ডিবাগিং করা জটিল হতে পারে কারণ প্রাইভেট মেথড এবং প্রপার্টি গুলো সরাসরি অ্যাক্সেস করা যায় না ।

**অ্যাবস্ট্রাকশন এবং এনক্যাপসুলেশন মধ্যে পার্থক্য কি ?**

-----------------------------------

**অ্যাবস্ট্রাকশন:** কনটেক্সট উপর বেস করে যখন কোনো ক্লাস ডিজাইন করি তখন ঐ ক্লাসে কি কি মেথড এবং প্রপার্টি থাকবে এবং কতটুকু রিপ্রেসেন্ট করবে বা কতটুকু এক্সপোস করবে এই ডিজাইন ফিলোসফি বা ডিজাইন আইডিয়ার হল অ্যাবস্ট্রাকশন ।

**এনক্যাপসুলেশন:** অ্যাবস্ট্রাকশন উপর বেস করে যখন কোনো ক্লাসটা ইমপ্লিমেন্টেশন করি তখন ঐ ক্লাসে কি কি মেথড এবং প্রপার্টি রিপ্রেসেন্ট করবো বা এক্সপোস করবো তা এনক্যাপসুলেশন মাধ্যমে করা হয় ।

**মূল পার্থক্য**

অ্যাবস্ট্রাকশন হল কী কাজ করতে হবে তার উপর ফোকাস করা, যেখানে এনক্যাপসুলেশন হল কিভাবে কাজটি বাস্তবায়িত হবে এবং ডেটা লুকিয়ে রাখা হবে তার উপর ফোকাস করা।

অ্যাবস্ট্রাকশন ব্যবহারকারীকে একটি সরল ইন্টারফেস প্রদান করে, যাতে অভ্যন্তরীণ জটিলতার সাথে ব্যবহারকারীকে পরিচিত হতে না হয়। অন্যদিকে, এনক্যাপসুলেশন ডেটা এবং মেথডগুলিকে একত্রিত করে এবং ডেটার অনাকাঙ্ক্ষিত অ্যাক্সেস রোধ করে।

সংক্ষেপে:

অ্যাবস্ট্রাকশন জটিলতা লুকিয়ে ফেলে, যখন এনক্যাপসুলেশন ডেটা এবং কোডকে একত্রিত করে।

অ্যাবস্ট্রাকশন ব্যবহারকারীদের জন্য জিনিসগুলিকে সহজ করে তোলে, যখন এনক্যাপসুলেশন কোডকে আরও নিরাপদ এবং রক্ষণাবেক্ষণযোগ্য করে তোলে।

-----------------------------------

কনক্রিট ক্লাস, অ্যাবস্ট্রাক্ট ক্লাস এবং ইন্টারফেস কি ?

-----------------------------------

কনক্রিট ক্লাস: নরমালি আমরা যে ক্লাস তৈরি করি তা হলো কনক্রিট ক্লাস । কনক্রিট ক্লাসে অবজেক্ট তৈরি করা যায় এবং সবগুলো মেথডের ইমপ্লিমেন্টেশন থাকে ।

অ্যাবস্ট্রাক্ট ক্লাস: যখন কোনো ক্লাসকে অ্যাবস্ট্রাক্ট কীওয়ার্ড দিয়ে ডিক্লেয়ার করা হয় তাকে অ্যাবস্ট্রাক্ট ক্লাস বলে । যখন আমরা কোনও ক্লাস ইমপ্লিমেন্টেশন করি তখন কখনও কখনও আমাদের এমন সিচুয়েশন মধ্যে পড়তে হয় ক্লাসের সবগুলো মেথডের ইমপ্লিমেন্টেশন করা সম্ভব হয় না । এই ক্ষেত্রে একটি ক্লাসকে অ্যাবস্ট্রাক্ট হিসাবে ঘোষণা করি । যখন কোনো ক্লাস অ্যাবস্ট্রাক্ট ক্লাসকে এক্সটেন্ড করে তখন উক্ত ক্লাস তার মত করে মেথডের ইমপ্লিমেন্টেশন করবে । ইনহেরিট করা জন্য extends কীওয়ার্ড ব্যবহার করা হয় । উদাহরণ, একাউন্ট ক্লাস, উইথড্র সিস্টেম ব্যাংক অনুযায়ী ভিন্ন ।

সীমাবদ্ধতাগুলো: ----------

১) অ্যাবস্ট্রাক্ট ক্লাসে অবজেক্ট তৈরি করা যায় না ।

২) অ্যাবস্ট্রাক্ট ক্লাসে abstract mehtod প্রাইভেট ডিক্লেয়ার করা যায় না

৪) অ্যাবস্ট্রাক্ট ক্লাসে অ্যাবস্ট্রাক্ট মেথড গুলো চাইল্ড ক্লাসে ডিক্লেয়ার করতে হয় ।

**ইন্টারফেস:**

যখন কোনো ক্লাসকে ইন্টারফেস কীওয়ার্ড দিয়ে ডিক্লেয়ার করা হয় তাকে ইন্টারফেস ক্লাস বলে । ইন্টারফেস হল একটি কনট্রাক্ট । যখন একটি ক্লাস ইন্টারফেসকে ইমপ্লিমেন্ট করবে, তখন উক্ত ক্লাস ইন্টারফেসের নির্দিষ্ট মেথডগুলি ইমপ্লিমেন্ট করবে । ইন্টারফেস ক্লাসে মেথডের ইমপ্লিমেন্টেশন থাকে না । Interface class কে অ্ন্য ক্লাস implement করতে হলে তার সকল মেথড কে implement কেরতে হয়। । ইনহেরিট করা জন্য implements কীওয়ার্ড ব্যবহার করা হয় । বড় কোনো ইন্টারফেস তৈরি করবো না, যাতে চাইল্ড ক্লাসে আননেসেসারি রেস্পন্সিবিলিটি লোড তৈরি হয় ।

**কয়েকটি গুরুত্বপূর্ণ পয়েন্ট: ----------**

১) ইন্টারফেস মেথড গুলো শুধুমাত্র signature (নাম, রিটার্ন টাইপ, প্যারামিটার) প্রোভাইড করে।

২) ইন্টারফেস ডিপেন্ডেন্সি ইনভার্শন প্রিন্সিপল অনুসরণ করতে সহায়তা করে। ক্লাসগুলো নির্দিষ্ট ক্লাসের উপর নির্ভর করে না, বরং ইন্টারফেসের উপর নির্ভর করে।

৩) ইন্টারফেস মূলত কোডের নির্দিষ্ট স্ট্রাকচার এবং বিহেভিয়ার নিশ্চিত করতে ব্যবহৃত হয়, যাতে বিভিন্ন ক্লাসের মধ্যে সামঞ্জস্য বজায় থাকে এবং নির্দিষ্ট কার্যকারিতা নিশ্চিত হয়।

ইন্টারফেস সীমাবদ্ধতাগুলো: ----------

১) ইন্টারফেস এর অবজেক্ট তৈরি করা যায় না ।

২) ইন্টারফেস এর সবগুলো মেথড বাই ডিফল্ট অ্যাবস্ট্রাক্ট ।

৩) ইন্টারফেস এর মেথড গুলোকে প্রাইভেট , প্রটেক্টেড ডিক্লেয়ার করা যায় না । ইন্টারফেসে প্রপার্টি থাকে না ।

৪) ইন্টারফেসকে ইমপ্লিমেন্ট করতে হয় । মাল্টিপল ইন্টারফেস ক্লাস ইমপ্লিমেন্ট করা যায় ।

| **ইন্টারফেসএবংঅ্যাবস্ট্রাক্টক্লাসেরমধ্যেপার্থক্য** | | |
| --- | --- | --- |
|  | **ইন্টারফেস** | **অ্যাবস্ট্রাক্টক্লাস** |
| কোড | অ্যাবস্ট্রাক্টমেথড, কনস্ট্যান্ট | অ্যাবস্ট্রাক্টমেথড, কনস্ট্যান্ট, কনক্রিটমেথড, কনক্রিটভ্যারিয়েবল |
| অ্যাক্সেসমডিফায়ার | public | public  private  protected |
| প্যারেন্টক্লাসেরসংখ্যা | একটিচাইল্ডক্লাসেরএকাধিকপ্যারেন্টক্লাসবাইন্টারফেসথাকতেপারে। | একটিচাইল্ডক্লাসেরশুধুমাত্রএকটিপ্যারেন্টক্লাসবাঅ্যাবস্ট্রাক্টথাকবে। |

**ইনহেরিটেন্স কাকে বলে ?**

Inheritance অর্থ হলো উত্তরাধিকার । যখন একটি class(parent) এর বৈশিষ্ট্য (methods & property) inherit করে অন্য class(child) এ ব্যবহার করে তাকে Inheritance বলে । Inherit করা জন্য extends keyword ব্যবহার করে ।

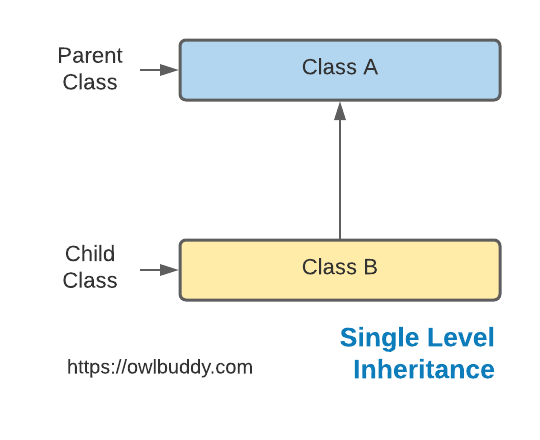
**ইনহেরিটেন্স আসলো দুইটা কারণে: ----------**

১) কোড রিইউজ করার জন্য ।

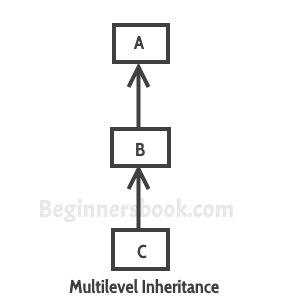
২) হায়ারার্কিক্যাল রিলেশন ইনহেরিটেন্স দিয়ে বোঝানোর সহজ ।

**ইনহেরিটেন্স এর প্রকারভেদ--**

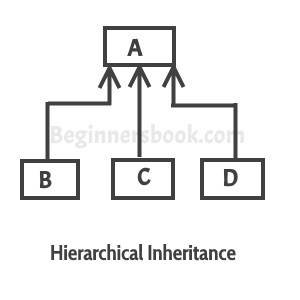
* Single Inheritance-যখন একটি ক্লাস শুধুমাত্র একটি ক্লাসকে ইনহেরিট করে তাকে সিঙ্গেল ইনহেরিটেন্স বলে ।



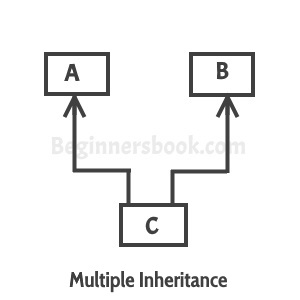
* **Multilevel Inheritance -** ধরি A, B, C তিন টা ক্লাস , যখন B ক্লাস A কে এবং C ক্লাস B কে ইনহেরিট করে তাকে মাল্টিলেভেল ইনহেরিটেন্স বলে ।



* **Hierarchical Inheritance-**যখন একটি ক্লাসের একাধিক চাইল্ড ক্লাস থাকে তাকে হায়ারার্কিক্যাল ইনহেরিটেন্স বলে । হায়ারার্কিক্যাল ইনহেরিটেন্স এর সীমাবদ্ধতা হলো ডায়মন্ড প্রবলেম হতে পারে।
* অসুবিধা: ----------
* ১) প্যারেন্ট ক্লাসে পাবলিক মেথড বা প্রপার্টি গুলো চাইল্ড ক্লাস চাইলোও প্রাইভেট করতে পারবে না ।
* ২) যেহেতু চাইল্ড ক্লাস প্যারেন্ট ক্লাস কে এক্সটেন্ড করে, তাই চাইল্ড ক্লাস চাইলে প্যারেন্ট ক্লাসে একসেসবলে মেথড এবং প্রপার্টি গুলো ওভাররাইট করতে পারে। যার ফলে এনক্যাপসুলেশনের এর প্রাইমারি কনসেপ্ট ব্রেক করে ।



* **Multiple Inheritance-যখন** একটি ক্লাস একাধিক ক্লাসকে ইনহেরিট করে তাকে মাল্টিপল ইনহেরিটেন্স বলে । PHP তে মাল্টিপল ইনহেরিটেন্স সাপোর্ট নয় । PHP তে মাল্টিপল ইনহেরিটেন্স এর সীমাবদ্ধতা দূর করার জন্য Trait ব্যবহার করা হয় । মাল্টিপল ইনহেরিটেন্স সাপোর্ট না করার কারণ হলো একাধিক প্যারেন্ট ক্লাসে যদি এই নাম মেথড থাকলে তাহলে চাইল্ড ক্লাস কোনটা মেথড ব্যবহার করবে কনফিউস ক্রিয়েট হয় ।



**এক্সেস মডিফায়ার কি ?**

এক্সেস মডিফায়ার (Access Modifier) হল একটি কনসেপ্ট যা ব্যবহার করে প্রোগ্রামে একটি ক্লাসের মেথড, এবং প্রপার্টির অ্যাক্সেস কন্ট্রোল করা হয়- এক্সেস মডিফায়ার নির্ধারণ করে কোন অংশগুলো বাইরের কোড থেকে অ্যাক্সেস করা যাবে এবং কোনগুলো যাবে না।

এক্সেস মডিফায়ার হল একটি কীওয়ার্ড যা একটি ক্লাসের মেথড বা প্রপার্টির এক্সেস নির্ধারণ করে ।

তিনটি ধরনের এক্সেস মডিফায়ার আছে: ----------

**Public:**

ক্লাসের মধ্যে যে সকল মেথড এবং প্রপার্টি কে অবজেক্ট ক্রিয়েট করে এবং চাইল্ড ক্লাসে এক্সেস দিতে চাইলে পাবলিক ডিক্লেয়ার করতে হয় ।একটি Public প্রপার্টি বা মেথড ক্লাসের বাইরে থেকে সরাসরি অ্যাক্সেস করা যায়। এটি ক্লাসের বাইরে থেকেও ব্যবহার করা যায়।

**Protected:**

একটি Protected প্রপার্টি বা মেথড শুধুমাত্র সেই ক্লাস এবং তার চাইল্ড (Inherited) ক্লাসের ভিতরেই অ্যাক্সেস করা যায়। ক্লাসের বাইরে থেকে অ্যাক্সেস করা যায় না।

**Private:**

একটি Private প্রপার্টি বা মেথড শুধুমাত্র সেই ক্লাসের ভেতরেই অ্যাক্সেস করা যায়। ক্লাসের বাইরে বা চাইল্ড ক্লাস থেকেও এটি অ্যাক্সেস করা যায় না। গেটার এবং সেটার মেথড ব্যবহার করে ভ্যালু গেট এবং সেট করা যায় ।

**Static, This, Self & Parent কীওয়ার্ড কেন ব্যবহার করা হয় ?**

**This:**

$this হলো সুডো ভ্যারিয়েবল, এটি কারেন্ট ক্লাসের অবজেক্টকে রেফার করে । নন-স্ট্যাটিক মেথড এবং প্রপার্টিকে ক্লাসের মধ্যে কল করার জন্য $this কীওয়ার্ড ব্যবহার করা হয় । PHP আপডেট ভার্সনে (৭.০) $this কিওয়ার্ড ব্যবহার করে স্ট্যাটিক মেথড এবং প্রপার্টিকে অ্যাক্সেস করা যায়, এক্ষেত্রে স্কোপ রিসোলিউশন অপারেটর(::) ব্যবহার করা হয় । যদিও এটি একটি ভালো প্র্যাকটিস নয় এবং PHP এর অন্য সংস্করণে এটি এরর জেনারেট করতে পারে ।

**Static:**

ক্লাসের মধ্যে যে সকল মেথড এবং প্রপার্টি অবজেক্ট ক্রিয়েট ছাড়া সরাসরি ক্লাস দিয়ে এক্সেস করার জন্য স্ট্যাটিক কীওয়ার্ড ব্যবহার করা হয় । স্ট্যাটিক মেথড এবং প্রপার্টিকে ক্লাসের ভিতরে অথবা বাহিরে কল করার জন্য স্কোপ রিসোলিউশন অপারেটর(::) ব্যবহার করা হয় । স্ট্যাটিক মেথড শুধু মাত্র স্ট্যাটিক প্রপার্টি এবং কনস্ট্যান্ট কে কল করতে পারবে ।

**Self & Parent & Static:**

সেলফ কীওয়ার্ড কারেন্ট ক্লাসকে এবং প্যারেন্ট কীওয়ার্ড চাইল্ড ক্লাস এর প্যারেন্ট ক্লাসকে রিপ্রেসেন্ট করে । সেলফ , প্যারেন্ট এবং স্ট্যাটিক কীওয়ার্ড দিয়ে ভ্যালু সেট করা যায় না । স্ট্যাটিক এবং নন-স্ট্যাটিক মেথড এবং স্ট্যাটিক প্রপার্টিকে কল করা যায় । নন-স্ট্যাটিক প্রপার্টিকে কল করা যায় না ।

**PHP তে ম্যাজিক মেথড গুলো কি কি ?**

ম্যাজিক মেথডগুলি ডাবল আন্ডারস্কোর (\_\_) দিয়ে শুরু হয় এবং সবগুলির একটা কমন বৈশিষ্ট্য আছে তাহল এই মেথডগুলি কখনও কল করতে হয়না বরং অটোমেটিক এক্সিকিউট হয় ।

**\_\_construct(), \_\_destruct(), \_\_call(), \_\_callStatic(), \_\_get(), \_\_set(), \_\_isset(), \_\_unset(), \_\_sleep(), \_\_wakeup(), \_\_toString(), \_\_invoke(), \_\_set\_state() এবং \_\_clone()** ফাংশনকে পিএইচপি ম্যাজিক মেথড বলা হয়।

**\_\_construct() & destruct():**

নতুন কোনো অবজেক্ট ক্রিয়েট করলে ক্লাসের মধ্যে যে মেথডটি

সবার আগে এক্সেকিউট হয়, তাকে \_\_\_কন্সট্রাক্টর বলে এবং অবজেক্টর কাজ শেষ হওয়ার পর ক্লাসের মধ্যে যে মেথডটি সবার পর এক্সেকিউট হয়, তাকে \_\_\_ডেস্ট্রাক্টর বলে ।

**\_\_call() & callStatic():**

যখন কোনো নন-এক্সিস্ট বা ইনএক্সেসিবল মেথডকে এক্সেস করার জন্য কল করা হয় তখন \_\_\_কল বা \_\_কালস্ট্যাটিক মেথড এক্সেকিউট হয় । এখানে নন-স্ট্যাটিক মেথডের জন্য \_\_\_কল মেথড এবং স্ট্যাটিক মেথডের জন্য \_\_কালস্ট্যাটিক মেথড এক্সেকিউট হয় । মেথডগুলো দুটি পারমিটার একসেপ্ট করে ফার্স্ট পারমিটের মেথডের নাম এবং সেকেন্ড পারমিটের আর্গুমেন্ট এর একটি অ্যারে গ্রহণ করে । \_\_\_কল এবং \_\_\_\_কালস্ট্যাটিক মেথড দুটি মেথড ওভারলোডিং এর সময় ব্যবহার করা হয় ।

**\_\_get() & set():**

যখন কোনো নন-এক্সিস্ট বা ইনএক্সেসিবল প্রপার্টির ভ্যালু এক্সেস করার জন্য কল করা হয় তখন \_\_গেট মেথড এবং প্রপার্টির ভ্যালু সেট করার জন্য কল করা হয় তখন \_\_সেট মেথড এক্সেকিউট হয় । \_\_সেট মেথড দুটি পারমিটার একসেপ্ট করে ফার্স্ট পারমিটের প্রপার্টির নাম এবং সেকেন্ড পারমিটের প্রপার্টির ভ্যালু একসেপ্ট করে । \_\_গেট মেথড শুধুমাত্র প্রপার্টির নাম একসেপ্ট করে ।

**\_\_invoke():**

যখন একটি অবজেক্টকে একটি ফাংশন হিসাবে বিবেচনা করা হয় তখন ইনভোক মেথড ব্যবহার করা হয় ।

**Traits(বৈশিষ্ট্য) কি?**

PHP তে multiple inheritance সাপোর্ট করেনা। এইসমস্যা solve করারজন্য Traits use করাহয়। Traits এরকোন object/instance create করাযায়না। একটি class multiple traits কে use করতেপারে।

নোট: যদি ট্রেইট, প্যারেন্ট ক্লাস এবং চাইল্ড ক্লাসে একই নামের মেথড থাকলে , তখন প্রথমে চাইল্ড ক্লাস , তারপর ট্রেইট , তারপর প্যারেন্ট ক্লাসে মেথড এক্সেকিউট হয় ।

**Namespaces**

নেমস্পেস মুলত ডিরেক্টরি বা ফাইল নির্দেশক। Same নামের ক্লাস বা ফাংশনকে বিভিন্ন namespaces এ define করা যায় এবং namespaces ব্যবহার করে element গুলো অ্যাক্সেস করা যায়।

**PHP তে Method Chaining কি?**

ধরি, একটা Class এ তিন টা মেথড আছে । এখন আমরা যদি এই তিনটি method কে আলাদা আলাদা ভাবে object দিয়ে call না করে একটি object দিয়ে call করার পদ্ধতিকে Method Chainin বলা হয় ।

**Public method:**

class Person{

private $name="";

private $age="";

public function setName($name="")

{

$this->name=$name;

return $this;

}

public function setAge($age="20")

{

$this->age=$age;

return $this;

}

public function getInfo()

{

echo "Hello, My name is ".$this->name." and my age is ".$this->age." years.";

}

}

$person = new Person();

$person->setName("Rubel")->setAge(50)->getInfo();

**Static method:**

class Person{

private static $name="";

private static $age="";

public static function setName($name="")

{

static::$name = $name;

return new static;

}

public static function setAge($age="20")

{

static::$age= $age;

return new static;

}

public static function getInfo()

{

echo "Hello, My name is ".static::$name." and my age is ".static::$age." years.";

}

}

$person = new Person();

$person::setName("Rubel")->setAge(33)->getInfo();

**ডিপেন্ডেন্সি ইনজেকশন কি ?**

যখন একটি ক্লাস তার প্রয়োজনীয় অবজেক্ট নিজে তৈরি করে না বরং অন্য ক্লাস তাকে প্রয়োজনীয় অবজেক্ট প্রদান করে তাকে ডিপেন্ডেন্সি ইনজেকশন বলে।

**কিভাবে অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং এ স্ট্যান্ডার্ড মানের কোড লেখা যায় ?**

১) OOP এর বেসিক কন্সেপ্ট গুলো বুঝা: ক্লাস, অবজেক্ট, ইনহেরিট্যান্স, পলিমরফিজম, এনক্যাপসুলেশন ইত্যাদি OOP এর মৌলিক ধারণা গুলো ভালোভাবে বুঝা ।

২) OOP এর ডিজাইন প্রিন্সিপাল গুলো শিখা: সিঙ্গেল রেসপনসিবিলিটি প্রিন্সিপল, ওপেন/ক্লোস্ড প্রিন্সিপলে, লিস্কভ সাবস্টিটিউশন প্রিন্সিপল, ইন্টারফেস সেগ্রেগেশন প্রিন্সিপল, ডিপেন্ডেন্সি ইনভার্শন প্রিন্সিপল ইত্যাদি শিখা এবং ব্যবহার করা ।

৩) OOP এর ডিজাইন প্যাটার্ন গুলো শিখা: সিঙ্গেলটন প্যাটার্ন, ফ্যাক্টরি প্যাটার্ন, অবজার্ভার প্যাটার্ন, ডেকরেট প্যাটার্ন ইত্যাদি শিখা এবং ব্যবহার করা ।

৪) প্রজেক্টের রিকুইরেমেন্ট এবং স্কোপ গুলো প্রপারলি বুঝা: আপনার প্রকল্পের রিকুইরেমেন্ট এবং স্কোপ সঠিকভাবে বুঝা এবং তার উপর ভিত্তি করে OOP তে ডিজাইন করা।

**ডিজাইন প্রিন্সিপল কি ?**

ডিজাইন প্রিন্সিপল হলো জেনারেল প্রিন্সিপল যা সফটওয়্যার ডিজাইন করার সময় ফলো করা উচিত ।

**কিছু জেনারেল ডিজাইন প্রিন্সিপল হলো:**

**১) SOLID Principle:**

**!) Single Responsibility Principle:** একটি ক্লাসে শুধুমাত্র একটি রেসপনসিবিলিটি থাকা উচিত। উদাহরণ, ধরি User নামে একটি ক্লাস আছে । এই ক্লাসে শুধুমাত্র ইউসার রিলেটেড কাজ করবে । যেমন নিবন্ধন, লগইন এবং পাসওয়ার্ড রিসেট । এটি অন্যান্য কাজগুলির জন্য রেস্পন্সিবল নয়, যেমন ইমেল পাঠানো, লগ ফাইল লেখা, অ্যাক্সেস কন্ট্রোল।

**!!) Open/Closed Principle:** ক্লাস এক্সটেনশনের জন্য ওপেন, কিন্তু মডিফিকেশনের জন্য ক্লোস থাকা উচিত । OCP প্রিন্সিপল অনুসরণ করে ক্লাস ডিজাইন করলে প্যারেন্ট ক্লাসের কোনো পরিবর্তন ছাড়াই চাইল্ড ক্লাস এক্সটেন্ড করে কাজ করতে পারবে।

**!!!) Liskov Substitution Principle:** একটি সাবক্লাস তার বেস ক্লাসের যেকোনো কনটেক্সট ব্যবহার করতে পারবে, কিন্তু ক্লাসের কোনো বিহেভিয়ার পরিবর্তন না করে।

**!v) Interface Segregation Principle:** একটি বড় ইন্টারফেসকে একাধিক ছোট এবং আরও নির্দিষ্ট ইন্টারফেসে বিভক্ত করার পরামর্শ দেয়। সহজ করে বলতে গেলে, একটি ইন্টারফেসে শুধুমাত্র সেই মেথডগুলো থাকা উচিত যা একসাথে ব্যবহার করা হয়।

**v) Dependency Inversion Principle**: হায়ার লেভেল ক্লাস লোয়ার লেভেল ক্লাস গুলোর উপর নির্ভর করবে না, বরং উভয়ই অ্যাবস্ট্রাক্ট ক্লাসের উপর নির্ভর করা উচিত। সাধারণত, হায়ার লেভেল ক্লাসগুলি অ্যাপ্লিকেশনের মূল লজিক পরিচালনা করে এবং লোয়ার লেভেল ক্লাসগুলি মৌলিক অপারেশন বা সার্ভিসগুলো প্রদান করে। লোয়ার লেভেল ক্লাস: EmailService

হায়ার লেভেল ক্লাস: UserService ।

**২) DRY(Don't Repeat Yourself) Principle:** এটি "নিজেকে পুনরাবৃত্তি করবেন না" এর একটি সংক্ষিপ্ত রূপ, যতটা সম্ভব কোড ডুপ্লিকেট এভোইড করে ৷

**৩) KISS(Keep It Stupid Simple) Principle:** এটি "কিপ ইট স্টুপিড সিম্পল" এর একটি সংক্ষিপ্ত রূপ, যতটা সম্ভব কোড সিম্পল রাখা ৷ এমনভাবে কোড লিখা যাতে কোড পড়া সহজ হয় এবং অপ্টিমাইজেশান বজায় রাখা ৷

**৪) Losse Coupling:** একটা মডিউল যদি অনেক গুলো মডিউলে উপর কম ডিপেন্ডেন্ট থাকে তাহলে তাকে লুস কাপলিং। কোনো একটি ইন্টারফেস বা কম মডিউলে উপর ডিপেন্ডেন্সি রাখবো।