<p>আমরা যারা অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং করি তাদের প্রায় সবার কাছেই শুরুর দিকে এব্‌স্ট্রাক ক্লাস এবং ইন্টারফেইস এর ধারনাটা বেশ বিভ্রান্তিকর হয়ে থাকে। এই লিখাটি পড়লে হয়তো এই দুইটা বিষয় সম্পর্কে ভাল একটা ধারণা পাওয়া যেতে পারে।</p>

<h2>ধারণা গত পার্থক্য</h2>

<p>প্রোগ্রামিং এর দিক থেকে বিবেচনা করলে এবস্ট্রাক্ট ক্লাস আর ইন্টার্ফেইস এর মধ্যে অনেক মিল পাওয়া যাবে। কিন্তু ধারণা গত দিক থেকে দুইটা সম্পুর্ণ ভিন্ন ভিন্ন বিষয়। ব্যাক্তিগত ভাবে আমার কাছে কোন একটা বিষয়ে ধারণাটাই(Concept) বেশি গুরুত্বপুর্ণ। আমরা বুঝে না বুঝে অনেক কিছুই করে ফেলি, কিন্তু যা করছি তার বিষয়ে সঠিক ধারণা থাকাটা খুবই জরুরী। </p>

<h2>এবস্ট্রাক্ট ক্লাসের ধারণা</h2>

<p>এবার কাজের কথায় আসি। মনেকরি, আমরা একটি গাড়ির প্রস্তুতকারক প্রতিষ্ঠান। আমরা গাড়ি বানাই। এখন আমাদের এক গ্রাহক এসে আমাদের বলল, আমাকে একটি গাড়ি বানিয়ে দিন। তো আমরা যেহেতু গাড়ির বানাই আমরা নিশ্চয় গাড়ি বানাতে পারব। কিন্তু ঘটনা হল, কেউ যদি আমাদেরকে গাড়ি বানাতে বলে তাহলে কি আমরা আসলেই গাড়ি বানাতে পারব? অবশ্যই পারব, কিন্তু প্রশ্ন হল কি গাড়ি বানাব? কেউ শুধু গাড়ি বানাতে বললেই আমরা গাড়ি বানাতে পারিনা কারণ তখনও আমাদের কাছে স্পষ্ট নয় যে আমরা কি গাড়ি বানাব। যখন কেউ গাড়ির কথা বলে তখন আমরা সবাই বুঝতে পারি যে গাড়ি অবাস্তব কিছু নয়, গাড়ি অস্তিত্ব আছে এই ব্যাপারে আমরা সবাই নিশ্চিত। কিন্তু সমস্যা হল, গাড়ি এমন একটি ধারণা যার আসলে বস্তুগত অস্তিত্ব নেই যতক্ষন পর্যন্ত আমরা বলব কি গাড়ি। এখানে গাড়ি একটি বিমূর্ত(এবস্ট্রাক্ট) সত্বা। আমরা সবাই জানি গাড়ি আছে কিন্তু আমরা কেউ গাড়ি দেখিনি, আমরা বাস দেখি, মাইক্রোবাস দেখি, ট্রাক দেখি, কিন্তু গাড়ি দেখি না, কারণ গাড়ি এক বিমূর্ত সত্বা। তার মানে আমাদের কাছে যখন কেউ বলে ট্রাক তখন আমাদের কাছে গাড়ির অস্তিত্ব আরও স্পষ্ট, এখনে ট্রাক বললে আমরা আরও নিশ্চিত ভাবে বুঝতে পারি আমাদের আসলে কি বানাতে হবে। আবার চিন্তা করি, যদি কেউ আমাদের এসে বলত, যে আমাকে একটি যানবাহন বানিয়ে দিন, তাহলে কি হত? যানবাহন কি বানানো যায়? কারণ কোন ধরনের যানবাহন, গাড়ি, নৌকা নাকি বিমান এই ব্যাপারে আমাদের কাছে কোন তথ্য নেই। এই ক্ষেত্রে যানবাহনও একটি বিমূর্ত সত্বা। যানবাহন এর চেয়ে আরেকটু নিশ্চিত হতে পারেছি যখন কেউ বলেছে গাড়ি বানাতে হবে। কিন্তু গাড়িও আমরা বানাতে পারি না কারণ গাড়ি নিজেও আরেকটা বিমূর্ত সত্বা। তবে এই উদাহরন থেকে আমরা বুঝতে পারছি যে যানবাহন বেশি বিমূর্ত কিন্তু গাড়ি তার তুলনায় কম বিমূর্ত, এর পর ট্রাক আরও কম, কিন্তবা আমরা বলতে পারি ট্রাক কোন বিমূর্ত সত্বা নয়, কারণ ট্রাকের সরাসরি অস্তিত্ব আছে। গাড়ি প্রস্তুককারক প্রতিষ্ঠানকে ট্রাক বানিয়ে দিতে বললে তারা কিছু একটা বানিয়ে দিতে পারবে। এই যে যানবাহন > গাড়ি > ট্রাক এভাবে বিমূর্ত সত্বার বিভিন্ন পর্যায়, একে আমরা অনেকে ভাব নিয়ে বলি লেয়ার অব এবস্ট্রাকশন(Lyer of Abstraction)। এখানে আমরা স্পষ্টভাবেই বুঝতে পারছি যে কোন বিমূর্ত্য সত্বার অস্তিত্ব তৈরি করা যায় না, প্রোগ্রামিং-এ ঠিক এই কারণেই এবস্ট্রাক্ট ক্লাসের কোন ইন্‌সটেন্স তৈরি করা যায় না। এখানে আবার আরেকটা বিষয় লক্ষ্যনীয় যে বিমূর্ত সত্বার কিছু কিছু বিষয় আবার বিমূর্ত না ও হতে পারে। যেমন গাড়িটা কিভাবে চলবে এটা আমরা জানি না, কিন্তু সব গাড়িই চলবে এটা আমরা জানি। তার মানে গাড়ির এই চলার যে ক্ষমতা এটা গাড়ির বিমূর্ত বৈশিষ্ট। আবার অন্যদিকে সব গাড়ির দরজা যদি একই ভাবে খোলার ব্যাবস্থা থাকে, তাহলে গাড়ির দরজা খোলার যে ক্ষমতা এটা তখন আর বিমূর্ত নয়। এখন কোনটা বিমূর্ত বৈশিষ্ট আর কোনটা নয় এটা আমরা আসলে সহজেই বুঝতে পারি। আমরা যদি চিন্তা করি যে, একটা গাড়ি গাড়ি হওয়ার জন্য কি কি প্রয়োজন? সেই সেই জিনিস গুলো যদি গাড়ির প্রকারভেদ অনুসারে ভিন্ন ভিন্ন প্রয়োগের হয় তাহলে ঐ বৈশিষ্টটি গাড়ির বিমূর্ত বৈশিষ্ট কিন্তু ট্রাকের জন্য সেগুলো আর বিমূর্ত বৈশিষ্ট না হওয়ার সম্ভাবনাই বেশি। কোন ক্লাস যদি ঐ এবস্ট্রাক্ট ক্লাস কে ইনহেরিটেন্স করে তাহলে তারা ঐ বিমূর্ত বৈশিষ্টকে সুনিশ্চিত ভাবে প্রয়োগ করবে, ফলে ঐ বৈশিষ্টগুলো আর বিমূর্ত থাকবে না। </p>

<h2>ইন্টারফেইসের ধারণা</h2>

<p>তা বিমূর্ত(এবস্ট্রাক্ট) সত্বা নিয়ে অনেক বক বক করা হল, এবার আসি ইন্টারফেইস আসলে কি? ইন্টারফেইস হলো একটা চুক্তি(Contract)। যারা যারা এই চুক্তিতে স্বাক্ষর করবে তাদের সবাইকে চুক্তির সব কিছু মেনে চলতে হবে। যেমন ধরুন, যেসব মোবাইল ফোন কোম্পানি USB Type C চার্জার দিয়ে চার্জ হয়, তাদের প্রত্যেকের আভ্যন্তরীন চার্জিং প্রক্রিয়া হয়ত ভিন্ন ভিন্ন, কিন্তু তারা যেহেতু USB Type C দিয়ে চার্জেবল, সেহেতু তাদের সবার মধ্যে USB Type C - পোর্ট থাকতে হবে। কারণ এখানে USB Type C Chargable হল একটা চুক্তি, এই চুক্তি যারা সই করবে তারাদের সব কোম্পানীর মোবাইলে অবশ্যই USB Type C পোর্ট থাকবে। এখন যে কোন USB Type C চার্জার দিয়ে ঐ মোবাইল চার্জ করা যাবে। চার্জারের জানার প্রয়োজন নেই ঐ মোবাইল কোন কোম্পানীর, চার্জার শুধু জানে যে এটা USB Type C চার্জেবল মোবাইল, সুতরাং এই চার্জার দিয়ে এই মোবাইল চার্জ হবে। অর্থাৎ চার্জার শুধু দেখতেছে কোম্পানী গুলো কি চুক্তি স্বাক্ষর করেছে। ঐ চুক্তি অনুসারে যা যা করার কথা সকল মোবাইল কোম্পানী তা তা প্রয়োগ(implement) করবে। ঠিক এই কারনে, আমরা ইন্টারফেইস কে (প্রয়োগ করি)Implement করি কিন্তু ইনহেরিট/এক্সটেন্ড করি না। এখানে আরেকটা বিষয় লক্ষনীয় ইন্টারফেইসের এই বৈশিষ্টের কারনে ইন্টাফেইস নামকরণ সাধারণত বিশেষন(Adjective)-বাচক শব্দ দিয়ে করার জন্য অনেকে পরামর্শ দিয়ে থাকেন। যেমন, Clonable, Comparable, Runnable, Serializable ইত্যাদি। অন্যদিকে ক্লাসের নামে স্বাধারণত বিশেষ্য(Noun) হয়ে থাকে, যেমন, Car, Bus, Truck, MicroBus ইত্যাদি। </p>

<p>এই ছিল এবস্ট্রাক্ট ক্লাস এবং ইন্টারফেইস-এর ধারণাগত(Conceptual) পার্থক্য। আমার ধারণা উপরের আলোচনার পড়ার পর যাদের কাছে মনে হত যে এবস্ট্রাক্ট ক্লাস আর ইন্টারফেইস একই জিনিস তাদের এই চিন্তার দূর হবে। এই দুইটা বিষয়ের মধ্যে ধারণাগত পার্থক্য যদি আমাদের কাছে স্পষ্ট হয়ে থাকে তবে আমি মনে করি পোগ্রাম লিখার ক্ষেত্রে যতই দুইটার আচরন একই রকম হোক না কেন, আমরা আর বিভ্রান্তির সাগরে হাবুডুবু খাব না। </p>

<h2>প্রায়োগিক পার্থক্য</h2>

<p>ধারনার বা চিন্তার জগত থেকে এবার আমরা একটু প্রায়োগিক দিয়ে যদি তাকাই, তাহলে দেখতে পাই, এবস্ট্রাক্ট ক্লাস-এর কোন ইন্সটেন্স তৈরি করা যায় না, কারণ এবস্ট্রাক্ট ধারনাটার কোন বাস্তব অস্তিত্ব নেই, যা আসলে আমাদের এবস্ট্রাক্ট্স এর ধারনা থেকেই আমরা বুঝতে পেরেছিলাম। এখন ঘটনা হল, আমরা তাহলে কোন একটা ক্লাসকে এবস্ট্রাক্ট কেন করব? কোন একটা ক্লাসকে আমরা এবস্ট্রাক্ট করব তখনই যখন আমরা দেখবে, যে ঐ ক্লাসটার এমন কিছু বৈশিষ্ট বা গুনাগুন আছে যার কিছু কিছু বৈশিষ্ট আমাদের কাছে স্পষ্ট আবার কিছু কিছু বৈশিষ্ট অস্পষ্ট। স্পষ্ট হবে যখন ঐ এবস্ট্রাক্ট ক্লাসের কনক্রিট(বিমূর্ত নয় এমন) ক্লাস দেখব। বেশিরভাগ প্রোগ্রামিং ভাষার নিয়ম অনুসারে, কোন ক্লাসের একটা মেথডও যদি এবস্ট্রাক্ট মেঠদ হয় তাহলে ঐ ক্লাসটি অবশ্যই এবস্ট্রাক্ট ক্লাস হবে, তবে কোন মেথড যদি এবস্ট্রাক্ট মেথড নাও হয় তবুও একটি ক্লাস এবস্ট্রাক্ট হতে পারে। নিচে আমরা PHP তে একটি এবস্ট্রাক্ট ক্লাসের উদাহরণ দেখতে পারি </p>

<pre><code>abstract class Car{

public $color;

abstract public function run();

public function openDoor(){

echo “All car’s door open same way”;

}

}</code></pre>

<pre><code>class Bus extends Car{

public function run(){

echo “Run like a bus”;

}

}</code></pre>

<pre><code>class Truck extends Car{

public function run(){

echo “Run like a truck”;

}

}</code></pre>

<p>এখানে, Car ক্লাসটি একটি এবস্ট্রাক্ট ক্লাস। এই ক্লাসটির কোন ইন্সটেন্স তৈরি করা যাবে না। এর মানে কে যদি লিখে $car = new Car() তাহলে আসলে প্রোগ্রামে Error দেখাবে। অন্যদিকে আমাদের Bus এবং Truck ক্লাস দুটি Car কে ইনহেরিট(extends) করেছে, তাই এরা অবশ্যই Car ক্লাসে run() নামে যে বিমূর্ত বৈশিষ্ট(Abstract method) ছিল তাকে এখানে স্পষ্টভাবে সঙ্গায়ীত(Concret definition) করতে হবে। ঠিক এই জায়গায় ইন্টারফেইসের সাথে এবস্ট্রাক্ট ক্লাসের মিল আছে, কারন এখানে ব্যাপারটা অনেকটা এমন হয়ে গেল যেন Bus এবং Truck ক্লাস দুটি কোন চুক্তি করেছে যে তাদের অবশ্যই run() মেথডকে প্রয়োগ(Implement) করতে হবে। যদি Bus ক্লাস run মেথড কে প্রয়োগ না করত তাহলে, অবশ্যই Bus ক্লাসটিও এবস্ট্রাক্ট ক্লাস হিসাবে ঘোষণা(Declare) দিতে হত। এই দুই শর্ত না মানলে, বেশিরভাগ প্রোগ্রামিং ভাষায়ই প্রোগ্রামটি Error দিবে। বেশিরভাগ প্রোগ্রামিং ভাষাই একটি ক্লাস একাধিক ক্লাসকে ইনহেরিট করতে পারে না।</p>

<p>এবার আমরা একটি ইন্টারফেইসের উদাহরণ দেখিঃ </p>

<pre><code>interface USBTypeCChargable{

public function chargeByUSBTypeC();

}</code></pre>

<pre><code>class SamsungMobile implements USBTypeCChargable(){

public function chargeByUSBTypeC(){

echo “Samsung Mobile is being charged…”;

}

}</code></pre>

<pre><code>class XiaomiMobile implemets USBTypeCChargable(){

public function chargeByUSBTypeC(){

echo “Xiaomi Mobile is being charged…”;

}

}</code></pre>

<p>এবার একটি Charger ক্লাস দেখে নেওয়া যাক যে চার্জার দিয়ে এই USB Type C Chargable মোবাইলগুলো চার্জ দেওয়া যাবে। </p>

<pre><code>class Charger{

public function charge(USBTypeCChargable $mobile){

$mobile->chargeByUSBTypeC();

}

}</code></pre>

</p>এই Charger ক্লাসটির charge() মেথডটি দেখে আমরা বুঝতে পারি, এই ক্লাসটির জানার প্রয়োজন নেই মোবাইলটি কি SamsungMobile নাকি XiaomiMobile কারণ এই চার্জারের শুধু দরকার যে মোবাইলটি সে চার্জ করবে ঐ মোবাইলটি USBTypeCChargable ইন্টারফেইসের চুক্তি সই করেছে কি না। যদি সই করে থাকে তবে অবশ্যই ঐ মোবাইল-এ chargeByUSBTypeC() মেথডটি থাকবে। তার মানে চার্জার এবং মোবাইলফোন Tightly Couple না অর্থাৎ ব্যাপারটা এমন নয় যে SumsungMobile শুধ Sumsung চার্জার দিয়েই চার্জ হবে। এখানে আরেকটা ব্যাপারে উল্লেখ্য একটি ক্লাস একাধিক ইন্টারফেইসকে প্রয়োগ(implement) করতে পারে। মানে ব্যাপারটা অনেকটা এমন যে একটা মোবাইল কোম্পানী অনেক গুলো চুক্তি করতে পারে। তারা চার্জারের জন্য যেমন চুক্তি করতে পারে তেমন হেয়ারফোনের জন্যও চুক্তি করতে পারে। এই ব্যাপারটা এবস্ট্রাক্ট ক্লাসের সাথে অমিল। পাশাপাশি, ইন্টারফেইসের কোন মেথডেই কোন প্রয়োগের কথা বলা থাকে না। মানে বলা যায় ইন্টারফেইসের সব গুলো বৈশিষ্টই(method/property) বিমূর্ত(এবস্ট্রাক্ট)। আর যেহেতু ইন্টারফেইসের সব গুলো মেথডই এবস্ট্রাক্ট তাই আলাদা ভাবে এবস্ট্রাক্ট লিখার প্রয়োজন নেই। </p>

<h2>সবশেষে</h2>

<p>এইটুকু বলব, আমার বিশ্বাস নতুনদের কাছে এখনো এবস্ট্রাক্ট ক্লাস এবং ইন্টারফেইসের ব্যাবহারের ক্ষেত্রে কিছু বিভ্রান্তি থেকে যেতে পারে যা দূর করার অন্যতম প্রধান উপায় বেশি বেশি অনুশীলন করা। যত বেশি প্রোগ্রাম লিখবে তবেই এই ব্যাপার গুলো পরিষ্কার হবে। তবে, আমরা যদি এখন ধারণাগত দিক থেকে বুঝতে পারি যে এবস্ট্রাক্ট ক্লাস এবং ইন্ট্রারফেইস আলাদা তাহলেই আমার এই লিখার স্বার্থকতা। পরবর্তি কোন লিখায় পলিমর্ফিজম নিয়ে আলোচনা করার চেষ্টা করব ইন-শা-আল্লাহ, কারণ আমি মনে করি পলিমর্ফিজম ব্যাবহার না করে এবস্ট্রাক্ট ক্লাস বা ইন্টারফেইসের আসল মজা বা প্রয়োজনীয়তা বুঝা যায় না। এই লিখায় একটি উদাহরণে ইতিমধ্যে আমরা পলিমর্ফিজম ব্যাবহার করেছি কিন্তু আসলে একই লিখায় এত কিছু ব্যাখ্যা দিয়ে আপনাদের আরও বিভ্রান্ত করতে চাই না তাই আমি পলিমর্ফিজমের ব্যাখ্যা এই লিখায় সচেতনভাবে এড়িয়ে গিয়েছি। </p>

<strong>কৃতজ্ঞতা স্বীকারঃ</strong>বন্ধু সুমন, গুরু ও বড় ভাই আনোয়ার হোসাইন, আরেক উস্তাদ ও বড় ভাই মিজবাহ আহ্‌সান বাহারাম এবং আমার ছাত্র ও ছোট ভাই ড্যাফোডেলিয়ান আবুল হাসনাত নাঈম কে ধন্যবাদ এই লিখার বানান ভুল গুলো সম্পর্কে অবগত করার জন্য।