পাঠ ৫.৩: এনক্যাপসুলেশান

আমরা এতোক্ষনে জেনে ফেলেছি যে, একটি অবজেক্ট হলো কতগুলো ডাটা এবং মেথড এর সমষ্টি। অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং এর আরেকটি খুবই গুরুত্বপূর্ণ বিষয় আছে, যা হচ্ছে, একটি ক্লাসের মধ্যে ডাটা গুলোকে লুকিয়ে রাখা এবং শুধুমাত্র মেথডের মাধ্যমে সেগুলোকে একসেস করতে দেওয়া। এর নাম হচ্ছে এনক্যাপসুলেশান(Encapsulation) । এর মাধ্যমে আমরা সব ডাটা গুলোকে ক্লাসের মধ্যে সিল করে একটা কেপসুলের মধ্যে রেখে দিই এবং সেগুলো শুধুমাত্র যেসব মেথড গুলোকে ট্রাস্ট করা যায়, তাদের মাধ্যমে একসেস করতে দিই।

তবে এই এতো প্রোটেকশান এর কারণ কি হতে পারে তা যদি একটু জেনে নিই শুরুতে তাহলে আমার মনে খুব ভাল হয় –

যারা অনেক লেখালেখি করে এমনকি যারা কোড লিখে তারাও জানে যে, একটা লেখা ততই ভাল হয় সেটাকে যত বেশি রি-রাইট করা হয়। আপনি যদি একটা কোড লিখে ফেলে রাখেন এবং কিছুদিন পরে আবার সেটি খুলে দেখেন- দেখা যাবে যে আপনি আরও একটি ভাল উপায় বের পেয়ে যাবেন সেই কোডটি লেখার। এটি সব সময়ই হয়। এই বার বার কোড চেঞ্জ করে নতুন করে লেখাকে বলা হয় রিফেক্টরিং(refactoring)। আমরা একটি কোডকে বার বার লিখে সেটাকে আরও বেশি কিভাবে সহজবোধ্য কোড লেখার চেষ্টা করি যাতে সেই কোডটি আরও ভালভাবে মেইনটেইন করা যায়।

কিন্তু এখানে একটি চিন্তার বিষয় হচ্ছে। আমরা জানি যে অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং এর মাধ্যমে আমরা যে সফটওয়্যার সিস্টেম তৈরি করি তাতে নানা রকম অসংখ্য অবজেক্ট থাকে যেখানে একটি অবজেক্ট আরেকটির সাথে তথ্য আদান প্রদান করে, একটি আরেকটির উপর নির্ভর করে কাজ করে থাকে। ধরা যাক, A একটি অবজেক্ট যার উপর B নির্ভর করে। ধরা যাক B এখানে কনজ্যুমার অবজেক্ট। এখন আমারা যদি A কে কোন রকম পরিবর্তন করতে চাই, তাহলে B আগের মতোই থাকতে চাইবে। এখানে দুটো অবজেক্ট হয়তো দুইজন ভিন্ন প্রোগ্রামার লিখে থাকতে পারে। সুতরাং একে অন্যের পরিবর্তন নিয়ে যাতে সমস্যায় পরতে না হয়, সেই ব্যবস্থা করতে হবে। আমরা অনেক সময় নানা রকম লাইব্রেরি ব্যবহার করতে হয় বিভিন্ন প্রজেক্টে এবং এগুলোর উপর নির্ভর করে করে আমাদের প্রজেক্ট দাড়িয়ে যায়। এই লাইব্রেরি গুলোর মাঝেই মাঝেই ভার্সন পরিবর্তন হয়। কিন্তু মজার ব্যপার হলো এগুলো পরিবর্তন হলেও আমাদের কোড নতুন করে লিখতে হয় না। আবার অন্যদিকে লাইব্রেরি যারা তৈরি করে তাদেরও এই কোড পরিবর্তনের স্বাধীনতা থাকা চাই, কিন্তু সক্ষেত্রে যাতে আমাদের প্রজেক্ট এর কোন সমস্যা যাতে না হয় সেটাও মনে রাখতে হবে।

তো এই সমস্যা সমাধানের একটি উপায় আছে। সেটি হলো- লাইব্রেরি কোড-এর যে মেথড গুলো আছে সেগুলো মোটেও রিমুভ করা যাবে না। কারণ আমরা যখন একটি লাইব্রেরির একটি নির্দিষ্ট ক্লাসের মেথড নিয়ে কাজ করবো, আমরা চাইবো না কোন ভাবেই আমাদের কোড ভেঙ্গে যাক। লাইব্রেরির প্রোগ্রামার সেই ক্লাস নিয়ে যা কিছু করতে পারবে, কিন্তু আমরা যে সব মেথড ব্যবহার করেছি সেগুলোকে মুছে ফেলতে পারবে না। তারপর ফিল্ড বা প্রোপ্রার্টিজ এর ক্ষেত্রেও লাইব্রেরি

যে লিখেছে সে কিভাবে জানবে যে কোন ফিল্ড বা প্রোপ্রার্টিজ গুলো আমরা আমাদের প্রজেক্ট এর ক্ষেত্রে একসেস করেছি? কোন ভাবেই জানার উপায় নেই। কারণ আমারা আমাদের কোড কিভাবে করেছি যা লাইব্রেরি যে লিখেছে তার জানার কথা নয়। কিন্তু যে প্রোগ্রামার লাইব্রেরি লিখেছে সে সবসময়ই চাইবে তার কোড এ নতুন কিছু এড করতে, আগের থেকে ভাল করা ইত্যাদি। এই সমস্যা সমাধানের জন্যে জাভা আমাদেরকে কতগুলো একেসেস স্পেসিফায়ার (access specifiers) দিয়ে থাকে, যার মাধ্যমে লাইব্রেরি প্রোগ্রামার ঠিক করতে পারে যে কোড এর কোন কোন অংশ গুলো আমরা যখন আমাদের প্রজেক্ট এ ব্যবহার করতে পারবো আর কোন কোন গুলো করতে পারবো না। এতে সুবিধা হচ্ছে, লাইব্রেরি প্রোগ্রামার সে সব অংশ গুলো আমাদেরকে ব্যবহার করতে দিচ্ছে, সেই অংশ গুলোতে ইচ্ছে মতো পরিবর্তন/পরিবর্ধন করতে পারবে কোন রকম চিন্তাভাবনা ছাড়া।

আমারা যখন একটা বড় সিস্টেমে কাজ করি আমাদের নানা রকম অবজেক্ট লিখতে হয়। একটি অবজেক্ট আরেকটি অবজেক্ট কে ব্যবহার করে। এই একেসেস প্রটেকশানের মাধ্যমে আমরা নির্ধারণ করে দিতে পারি যে একটি নির্দিষ্ট অবজেক্ট এর কোন অংশ গুলো অন্য অবজেক্ট ব্যভহার করতে পারবে ,আর কোন গুলো পারবে না। এতে উপরের সমস্যার সমাধান হয়ে যায়। এছাড়াও আরেকটি ব্যাপার হয়। আমরা যখন কোন একটি ক্লাস নিয়ে কাজ করতে যাবো, সেই অবজেক্ট-এ হাজার লাইন কোড থাকে পারে। পুরটা একেবারে দেখতে গেলে আমরা হয়তো কনফিউজড হয়ে যাবো কিংবা খুব কমপ্লেক্স কোড হলে বুঝতে অসুবিধা হতে পারে। কিন্তু সেই কোড যদি এমন ভাবে করা থাকে যেখানে অল্প অংশ আমাদের ব্যবহারের জন্যে অপেন করা থাকে, বাকি গুলো হাইড করা যাকে তাহলে আমরা যে অংশটুকু হাইড করা সেই অংশ নিয়ে চিন্তা করতে হবে না। এই কোড হাইড করার ঘটনাকে অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং এর ভাষায় এনক্যাপসুলেশান(Encapsulation) বলা হয়।

জাভাতে তিনটি একেসে কন্ট্রোল করার জন্যে তিনটি কি ওয়ার্ড আছে। সেগুলো হলে- `Public`, `protected` এবং `private` এখন আমরা বিভিন্ন রকম একসেস কন্ট্রোল দেখবো-

Default Access

এর মানে হচ্ছে আমরা যদি কোন কি-ওয়ার্ড ব্যবহার না করি তাহলে সেটি Default Access আর মাঝে পরে। কোন ক্লাস এর ভেরিয়েবল বা মেথড এর আগে যদি কোন একসেস মডিফায়ার না থাকে তাহলে সেই ক্লাসটি যে প্যাকেজের মধ্যে আছে সেই প্যাকেজ এর সব ক্লাস থেকে একসেস করা যাবে।

```
package bd.com.howtocode.java;

import java.util.Random;

public class HelloWorld {
    String version = "2.56";
```

```
    int getRandomInt() {
        return new Random().nextInt();
    }
}
```

এই ক্লাসের ভেরিয়েবল version এবং getRandomInt() মেথড কে bd.com.howtocode.java এই প্যাকেজ এর সকল ক্লাস একসেস করতে পারবে।

Private Access Modifier - `private`:

কোন ক্লাসের মেথড, ভেরিয়েবল, কনস্ট্রাকটর এর আগে যদি private কিওয়ার্ড থাকে তাহলে সেগুলোকে সেই ক্লাস ছাড়া অন্য কোন ক্লাস একসেস করতে পারবে না।

উদাহরণ-

```
    package bd.com.howtocode.java;

    public class User {
        private String name;
        private String emailAddress;

        public String getName() {
            return name;
        }

        public void setName(String name) {
            this.name = name;
        }

        public String getEmailAddress() {
            return emailAddress;
        }

        public void setEmailAddress(String emailAddress) {
            this.emailAddress = emailAddress;
        }
    }
```

এই ক্লাসের এর ভেরিয়েবল `name` এবং `emailAddress` কে কোন ভাবেই অন্য কোন ক্লাস থেকে একসেস করা যাবে না। কিন্তু আমরা যদি এদের কে একসেস করতে চাই তাহলে একসেসর মেথড ব্যাবহার করতে পারি।

Public Access Modifier - `public`:

কোন ক্লাসের মেথড, ভেরিয়েবল, কনস্ট্রাকটর এর আগে যদি public কিওয়ার্ড থাকে তাহলে সেগুলোকে যে কোন ক্লাস থেকে একসেস করা যায়।

```
public class Milk{
    public void swirl(boolean clockwise) {
        System.out.println("Swirling Milk");
    }
```

```
}
```

Protected Access Modifier - `protected`:

কোন ক্লাসের মেথড, ভেরিয়েবল, কনস্ট্রাকটর এর আগে যদি protected কিওয়ার্ড থাকে তাহলে সেগুলোকে অন্য প্যাকেজ থেকে সেই ক্লাসের সাব ক্লাস একসেস করতে পারবে আর নিজের প্যাকেজ এর সবাই একসেস করতে পারবে।

```
class AudioPlayer {
   protected boolean openSpeaker(Speaker sp) {
      // implementation details
   }
}
```

একসেস লেভেল একটি টেবিলি -

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N